Co-evolving protein sites: their identification using novel, highly-parallel algorithms, and their use in classifying hazardous genetic mutations

A thesis submitted in partial fulfilment
of the requirement for the degree of Doctor of Philosophy

Louise Knight

2017

Cardiff University
School of Computer Science & Informatics

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed . . . . . (candidate)

Date . . . . . .

Statement 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.

Signed . . . . . (candidate)

Date . . . . . .

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed . . . . . (candidate)

Date . . . . . .

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed . . . . . (candidate)

Date . . . . . .

Abstract

Algorithms for detecting molecular co-evolution have until now been applied only to individual protein families, but not to the human proteome. Linked to this is the problem that performing the computations for identifying co-evolving sites in the human proteome would take a prohibitively long time using the serial algorithms already in use. In addition, co-evolving sites have not been pursued as a possible way of classifying mutations according to their likelihood to cause disease. The main contributions of this thesis are as follows: identification of three suitable methods for detecting molecular co-evolution by comparing the performance of published state-of-the-art methods on simulated data; implementation of these methods in the parallel architecture CUDA, and evaluation of these methods' performance in comparison to serial implementations of the same methods; and identification of co-evolving sites across the entire human proteome, and analysis of these sites according to what is already known about disease-causing mutations. Beyond demonstrating the effectiveness of CUDA for implementing molecular co-evolution detection algorithms, we derive insights into techniques for efficient implementation of algorithms in CUDA (particularly algorithms which require tree-based structures, such as parametric methods), and our results provide preliminary insights into the relationship between co-evolving sites and mutation pathogenicity.

Acknowledgements

First I wish to thank my supervisors, Dr. Andrew Jones, Dr. Christine Mumford, and Dr. Andrew Pocklington, for their invaluable advice and encouragement throughout my PhD, and also particularly towards the end, when everything was coming together. I would like to express my gratitude for your constructive feedback on the numerous drafts this thesis has gone through. Again thank you for giving me this opportunity in the first place; it has been a privilege to learn and to grow my knowledge and experience in this fascinating field.

There are many members of staff in the School of Computer Science & Informatics that I have interacted with and who have helped me in various ways over the years; thanks go to you all.

I must thank my fellow research students in the School, not just for the numerous conversations about research, but also for providing a welcome distraction when necessary. If I start naming all of you, I risk missing someone out; there are truly too many of you to name.

I give thanks to my parents, John and Jean, for always encouraging me to go as far as I can, not just in my PhD, but throughout my whole life. It is through your constant support from the very beginning that I gained confidence in my abilities, without which I would never have got this far; you have always been my biggest cheerleaders. Thank you also for reminding me to take breaks (and ensuring I actually take those breaks!). Really, there is more to thank you for than I can actually write here; thank you.

Finally, thanks go to Ivan; although we unfortunately met when I had just over 3 months to go until submission, you implicitly understood the importance of this work to me and gave me the time, space, and support to finish everything up, for which I am very grateful.

Contents

List of Tables

List of Figures

Acronyms

ALF Artificial Life Framework.

ANOVA Analysis Of Variance.

APC Average Product Correction.

APOD Assess, Parallelise, Optimise, Deploy.

ARCCA Advanced Research Computing at Cardiff.

BLAST Basic Local Alignment Search Tool.

BLOSUM Blocks Substitution Matrix.

BMM Bayesian Mutational Mapping.

CAPS Co-evolution Analysis using Protein Sequences.

CASP Critical Assessment of protein Structure Prediction.

CASS Course-grained Artificial Sequence Simulator.

CPU Central Processing Unit.

CUDA Compute Unified Device Architecture.

DCA Direct-Coupling Analysis.

DI Direct Information.

DNA Deoxyribonucleic Acid.

ELSC Explicit Likelihood of Subset Co-variation.

EMBL European Molecular Biology Laboratory.

FASTA Fast-All.

GERP Genomic Evolutionary Rate Profiling.

GNU GNU's Not Unix.

GPGPU General Purpose Graphics Processing Unit.

GPU Graphics Processing Unit.

JTT Jones-Taylor-Thornton.

McBASC McLachlan-Based Substitution Correlation.

MI Mutual Information.

ML Maximum Likelihood.

MLA Maximum Likelihood Approximation.

MSA Multiple Sequence Alignment.

MTREV Mitochondrial Reversible.

NCBI National Center for Biotechnology Information.

OMES Observed Minus Expected Squared.

PAM Point Accepted Mutation.

PHYLIP Phylogeny Inference Package.

PIM Percentage Identity Matrix.

PSIC Position-Specific Independent Counts.

PSICOV Protein Sparse Inverse Co-variance.

RNA Ribonucleic Acid.

SCA Statistical Coupling Analysis.

SIFT Sorting Intolerant From Tolerant.

SIMD Single Instruction Multiple Data.

SNP Single Nucleotide Polymorphism.

SPRT Swiss-Prot TrEMBL.

TCS Transitive Consistency Score.

TrEMBL Translation of EMBL nucleotide sequence database.

UAA Unique Amino Acids.

UCSC University of California, Santa Cruz.

UPGMA Unweighted Pair-Group Method with Arithmetic mean.

XML Extensible Markup Language.

Chapter 1

Introduction

Proteins are the building blocks of the body, coded for by genes. Changes to genes (called mutations) can change the corresponding protein produced, which could be associated with disease. There are several approaches to finding mutations which could be associated with disease; one which has not largely been pursued until this point is the idea of molecular co-evolution, which is where these mutations are fixed at different positions within a protein, seemingly simultaneously. There has not been, at time of writing, a method for detecting co-evolving sites which can be run in a feasible amount of time on all human proteins, while also being a well-performing algorithm from a biological point of view. This thesis shows that the programming language CUDA in combination with NVIDIA graphics cards can be used to perform this once-impossible task.

In this chapter we outline the background and hypothesis to the problem we are working on, to be given in more detail in the next chapter. We then introduce the thesis contributions, and the structure for the rest of this thesis.

1.1 Overview

DNA (Deoxyribonucleic Acid) is used to make RNA (Ribonucleic Acid) in the process of transcription (more detail on this process is given in Chapter 2). Proteins are then made from RNA. Proteins have many uses in the body, such as being messengers, antibodies, etc. In addition, the body itself is made substantially from protein. Changes to the DNA for coding a particular protein can cause changes to the protein itself, such as altering its structure or function. DNA, RNA, and protein sequences can all be represented as strings of characters.

A *Multiple Sequence Alignment* (MSA) is the result of taking multiple protein sequences which are related (i.e. from related species) and aligning them in such a way that we can see the similarities between them. When we look at an alignment, we can sometimes observe patterns where, at the rows where a particular amino acid (protein character) X appears at one column in the alignment, another amino acid Y (here we are using X and Y as place-holders) appears in another column (within the same row/sequence). (At other rows we could have cases where, whenever another amino acid A appears in the first column, an amino acid B appears in the second.) When this happens for a lot of rows within the alignment, it is called covariation. It is theorised that covariation is caused by molecular co-evolution, which can be defined as 'the direct physical effect of one sequence position on the other' [177, pp. 341]. Because of this, we can use multiple sequence alignments to find co-evolving positions. Molecular co-evolution can occur in such a way that it allows the evolution of new structures and functions within a particular protein, or to fix "errors" which occur.

It is believed that co-evolving sites are important to a protein's function and structure because of the actual process behind co-evolution. Pairs, or groups, of positions appear to evolve together, such that a change at one site can be compensated for by one or more changes at other sites. If the initial change did not alter the structure or function of the protein, then it would not need to be "corrected" in such a way. If a site is important (regardless of whether it is co-evolving with any other site or not), then changes at that site may cause the protein's function or structure to change, and it is this that can be associated with disease. These changes are called mutations. There are plenty of methods for predicting the possible pathogenicity[1] of mutations in humans, such as SIFT and PolyPhen (more detail on these appears in Chapter 8), but these have not really focused on the concept of molecular co-evolution, or amino acid covariation.

There have been several reviews of various methods for detecting molecular co-evolution (and countless more papers introducing new methods), but none of these have been applied, at the time of writing, to the entire human proteome. We believe this is due, at least in part, to the fact that the algorithms that have been developed for detecting molecular co-evolution thus far are (relatively) slow. Here we take the problem and apply high performance computing (HPC) to it. NVIDIA brand graphics cards are a relatively inexpensive way of utilising the massive amounts of computing power now available to us.

[1] If something is pathogenic, it causes disease.

The kind of parallelisation used in NVIDIA cards allows one to perform the same computer instruction on lots of data at once; this means we could search for co-evolving sites across a larger data set quicker than has been done before. CUDA is the programming language used to manipulate data on NVIDIA graphics cards. At the time of writing, only one paper has been identified [191] which uses graphics cards to identify co-evolving sites in a small data set. Due to CUDA's specific architecture, only certain kinds of problems can be effectively mapped to CUDA. These are problems where the same set of instructions is performed on large data sets, which means that work needs to be done to map a sequential algorithm to the CUDA architecture. These are all considerations that will be taken into account when choosing the method(s) to implement using CUDA.

1.2 Hypothesis

It cannot be ignored that this thesis is inherently multi-disciplinary, with the disciplines in question being Computer Science and Medicine. Due to this, we have a two-part hypothesis:

Part 1: Until this point, it has not been possible to find co-evolving sites in the entire human proteome in a feasible amount of time. We propose that translating a method for detecting co-evolving sites onto the CUDA architecture can allow us to solve this problem effectively. Part 2: Applying such a method to the human proteome shall allow us to test the hypothesis: measures of co-evolution can be used to identify functionally important sites that by their very nature cannot be found by looking at evolutionary conservation.

1.3 Thesis contributions

The contributions expected to be made through this thesis are:

- identification of a method (or a set of methods) for detecting molecular co-evolution which is suitable for use on the human proteome
- identification of a novel method for allowing tree-based data structures to be mapped to the CUDA architecture – this particular method is useful for when the tree only needs to be parsed (read) once
- improved runtime performance of the chosen method(s) which allows us to look at larger data sets than has been done before, in a "reasonable" amount of time

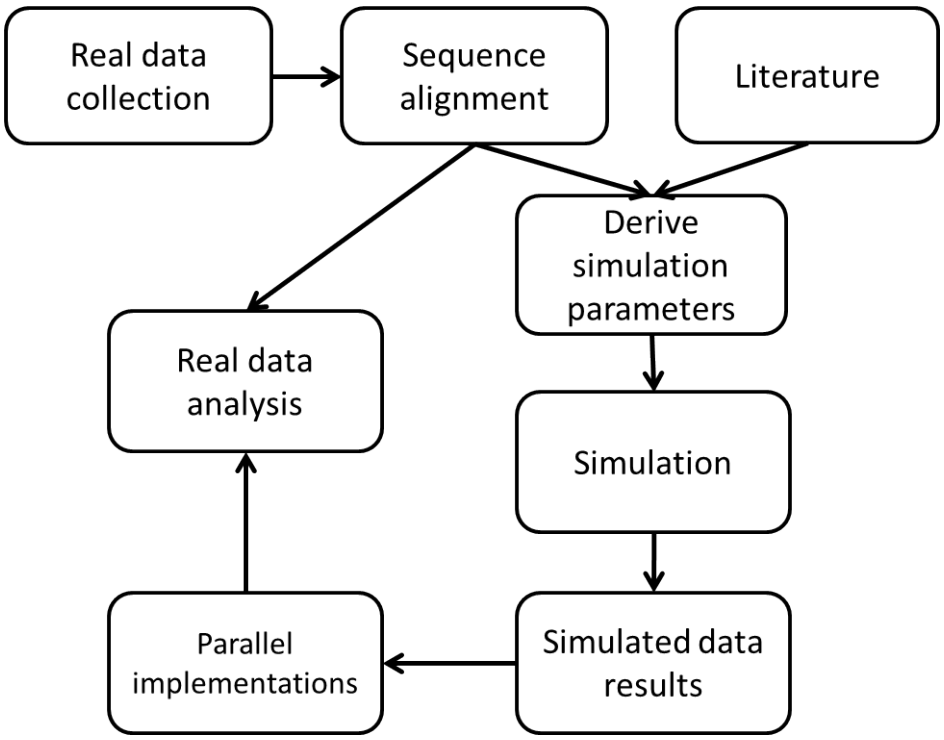

Figure 1.1: The steps involved in the completion of this thesis. Each step is linked to the relevant chapter in the main body of the text.

- make recommendations for how methods similar to those implemented in this thesis may be parallelised
- identification of co-evolving sites across the entire human proteome, and analysis of these sites according to what is already known about disease-causing mutations

These expected contributions draw from the hypothesis set out above, and have been designed in such a way that they allow us to test our hypothesis. In Chapter 9, we shall iterate through these contributions and show how they were fulfilled.

1.4 Thesis structure

Figure 1.1 shows the experimental approach used in this thesis. Within each chapter, we shall show which boxes of the diagram are applicable. Now we shall describe the contents of the rest of the chapters in this thesis and link these to Figure 1.1:

- Chapter 2 introduces background knowledge essential to understanding the rest of this thesis, in particular biological knowledge and an introduction to parallel computing.

- Chapter 3 talks through the methods which already exist for detecting molecular co-evolution, and compares them, detailing the choice of methods for comparison.

- Chapter 4 details the data collection methodology used for collecting real data ("Real data collection") and for constructing alignments from these sequences ("Sequence alignment"). The "real data" comprises those alignments of proteins from the human proteome with sequences from other, related, species, such that we can see the similarities between the proteins across the species. This data set is used later on in the thesis to find co-evolving sites in the human proteome.

- Chapter 5 describes how parameters were taken from the real data collected in the previous chapter and also from reading the literature ("Derive simulation parameters"), and these parameters are then used to simulate data sets ("Simulation"). The simulated data is used to compare a set of co-evolution detection methods according to their performance in Chapter 6. The reason why simulated data is used for this purpose rather than real data is so that we can control a lot of the parameters that may affect the performance of the methods being compared. In addition, we shall know the "ground truth" concerning where the co-evolving sites are. Simulation parameters are defined so that the simulated data is as close as possible to the real data.

- Chapter 6 compares the methods chosen in Chapter 3 by their performance on the simulated data ("Simulated data results").

- The results of the previous chapter are used to pick methods for parallelisation in Chapter 7 ("Parallel implementations").

- Chapter 8 looks at the results of running the parallelised methods on the real data set collected earlier ("Real data analysis").

- Chapter 9 concludes the thesis, linking the thesis contributions to the relevant chapters, and discusses this work's limitations and future work.

1.5 Summary

In this chapter we introduced briefly the problem and hypothesis approached in this thesis, as well as the thesis contributions and structure. In the next chapter we shall give more detailed background information to the problem.

Chapter 2

Background

This chapter presents essential background information concerning concepts used in the rest of the thesis. First we explain how proteins are made from DNA and RNA, and how changes to DNA can cause changes to the corresponding protein, which may cause disease. After discussing how variation in mutation rates among sites can be modelled and how some mutations can be more "acceptable" than others, we introduce the concept of species, and of phylogenetic trees, which attempt to explain how different species are related. Phylogenetic trees are used directly in some co-evolution detection methods (parametric methods), so we need a standard format for representing them so that they can be read into a computer program (e.g. Newick format); we also need to be able to construct phylogenetic trees from a set of sequences. Parametric co-evolution detection methods also require the ancestral sequences (those protein sequences of the ancestors to the species that are alive today), so we describe methods for reconstruction of ancestral sequences. All co-evolution detection methods (parametric and non-parametric) use multiple sequence alignments (MSAs) as input, so these are defined, along with examples of common alignment formats (e.g. FASTA (Fast-All), PHYLIP (Phylogeny Inference Package)). Co-evolution itself is defined (methods for detecting this in data are covered in the next chapter), along with another measure used to find "important" positions within proteins, conservation. Parallel computing is then broadly described, along with the specific architecture that will be used in this thesis, CUDA. The results of running the parallel co-evolution detection methods on the human proteome shall be used to test hypotheses concerning complex brain disorders, so in this chapter we finish with a brief overview of the genetics of complex brain disorders.

Figure 2.1: The structure of a DNA double helix. Image is in the Public Domain [7].

2.1 How proteins are made

This section describes the process by which proteins are produced. Proteins have many important roles in the human body; they act as messengers, antibodies, building blocks of the body itself, etc. The reader is referred to [203] for more detail.

First we describe what DNA, RNA, and proteins are, and how they are formed. We then describe the different levels of structure of a protein, and how changes to a DNA sequence can cause disease.

DNA (Deoxyribonucleic Acid) can be thought of as a string of characters, the allowed alphabet being the set $\{A, C, G, T\}$ (A = Adenine, C = Cytosine, G = Guanine, T = Thymine). DNA is used to make RNA (Ribonucleic Acid) in the process of transcription. RNA can also be thought of as a string of characters, where each T in DNA is replaced by a U (Uracil) during transcription. The units used to make DNA (or RNA) are called *nucleotides*, and each has different biological properties.

DNA molecules have a double helix structure (see Figure 2.1). They are made up of two strands of DNA, which in this thesis we choose to call the *sense* and *antisense* strand (other sources use the terms "coding" and "anti-coding" but these can be confusing). The sense strand tells us what the resulting RNA molecule will look like after transcription (replacing Ts with Us). The antisense strand is the one used in transcription. If the character in the

	U	C	A	G
U	UUU F UUC F UUA L UUG L	UCU S UCC S UCA S UCG S	UAU Y UAC Y UAA Stop UAG Stop	UGU C UGC C UGA Stop UGG W
C	CUU L CUC L CUA L CUG L	CCU P CCC P CCA P CCG P	CAU H CAC H CAA Q CAG Q	CGU R CGC R CGA R CGG R
A	AUU I AUC I AUA I AUG M	ACU T ACC T ACA T ACG T	AAU N AAC N AAA K AAG K	AGU S AGC S AGA R AGG R
G	GUU V GUC V GUA V GUG V	GCU A GCC A GCA A GCG A	GAU D GAC D GAA E GAG E	GGU G GGC G GGA G GGG G

Figure 2.2: A table showing how every group of three RNA nucleotides maps to an amino acid. "Stop" indicates a stop codon, which signals that we should stop translation here.

sense strand is an A, the corresponding character in the antisense strand is a T (and vice versa). C and G are also paired in this way. These complementary base pairs are held together using hydrogen bonds.

Every group of three characters in the RNA string (called a *codon*) is translated to one amino acid, as seen in Figure 2.2. See Figure 2.3 for an example of how DNA is used to make RNA, and how RNA is then used to make proteins. The allowed alphabet of amino acids is the set $\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ (A = Alanine, C = Cysteine, D = Aspartic Acid, E = Glutamic Acid, F = Phenylalanine, G = Glycine, H

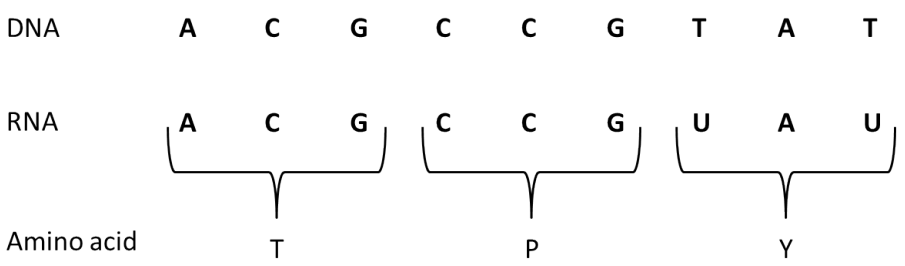

Figure 2.3: The only difference between a DNA sequence and the corresponding RNA sequence is the changing of T's to U's. Every group of three RNA bases is used to make one amino acid.

Figure 2.4: Graphical representation of an α-helix. Image is in the Public Domain [13].

Figure 2.5: Graphical representation of a β-sheet. Image is in the Public Domain [200].

= Histidine, I = Isoleucine, K = Lysine, L = Leucine, M = Methionine, N = Asparagine, P = Proline, Q = Glutamine, R = Arginine, S = Serine, T = Threonine, V = Valine, W = Tryptophan, Y = Tyrosine). Each amino acid has different biological properties.

Looking at the bigger picture, a *genome* can be thought of as the collection of "instructions" to make a species (species are defined more thoroughly in Section 2.3, but it is sufficient to say for now that humans, domestic cats, and domestic dogs are examples of species). The *gene* is the unit of information in a genome, and is made of DNA. A gene is made up not only of *protein-coding regions* (those parts which code for a protein), but also *non-coding regions*, which do not code for a protein, but have other uses, for example flagging whether the gene's sequence is going to be transcribed or not [21].

Proteins have different "levels" of structure. The string of amino acids is the *primary structure* of a protein. This "string" will fold into a three-dimensional structure (the *tertiary structure*). The tertiary structure is made up of smaller subunits, called *secondary structures*. Although the tertiary (3D) structure of a protein will inevitably differ between proteins, the secondary structures that make up a protein can be found in many different proteins [203]. Two of the main kinds of secondary structures are α-helices (Figure 2.4) and β-sheets (Figure 2.5). The sequence of amino acids which make up a protein influences that protein's structure and function.

2.2 Mutations

Now that we know what happens in the construction of proteins when things go right, we must consider what happens when things go wrong. A *mutation* is a change to a base within a DNA sequence. Mutations in the DNA sequence can cause changes to the corresponding amino acid sequence, due to the fact that the amino acid sequence depends on the DNA sequence. Because each amino acid has differing physicochemical properties, such changes could change the protein's structure or function, and because proteins have many important roles in the body, these changes could be associated with disease. For example, say we had a protein that gained a mutation, which caused the corresponding amino acid to change from being one that is "large" (by some definition) to one that is "small". Because the amino acids in a protein affect how that protein folds into its three-dimensional structure (and the way the protein folds directly affects its function), this mutation could cause a change in the structure of the corresponding protein, either making the folded protein unstable, or changing or otherwise damaging that protein's intended function. Such a change could cause disease. On the other hand, mutations can actually be beneficial; they could increase resistance to some hazard, for example. In addition, it all depends on the perspective from which a mutation's effect is viewed – a bacterium that gained a mutation and hence became more resistant to antibiotics is great for the bacterium, but bad for humans. Although we take into account that indeed mutations could positively impact an organism, in this thesis, we are concerned with those mutations which can cause disease in humans specifically.

Other changes that can occur in DNA sequences through evolution include *insertions* and *deletions*. Insertions are where a nucleotide/nucleotides are inserted into a DNA sequence, and deletions are where a nucleotide/nucleotides are deleted from a DNA sequence.

Having introduced the concept of mutations, we now discuss how differences in the mutation rates of sites in a protein can be modelled using the Gamma distribution, and then how substitution matrices describe whether, in general, a particular mutation tends to be "accepted" or not.

2.2.1 Gamma distribution to model mutation rate

The *mutation rate* of a (nucleotide) site is the number of times that site undergoes mutation per unit time. The mutation rate varies among different sites; some sites will have a higher rate of mutation than others. To model this,

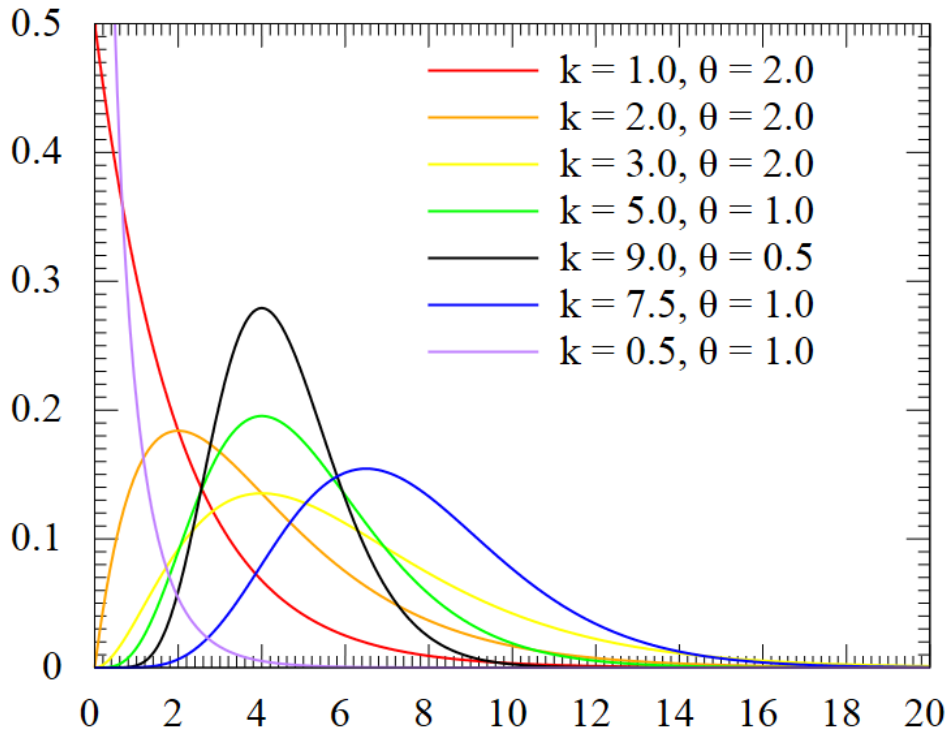

Figure 2.6: The probability density function of the Gamma distribution, showing the distribution for different values of k and θ. Figure obtained from [11] (available under Creative Commons license; see citation for details).

we can use the Gamma distribution [201]. This distribution has two parameters, α and β (in Figure 2.6, these parameters are given by k and θ, respectively). Varying these parameters can give us different distribution shapes; for example, for α (k) = 1.0, and β (θ) = 2.0, this tells us that most sites have low mutation rates, and very few have high rates. The simulation program used in this thesis, CS-PSeq-Gen, uses the continuous-rate Gamma distribution, and only requires the specification of the α parameter. (This simulation program is discussed further in Chapter 5.)

2.2.2 Amino acid similarity matrices

The book Understanding Bioinformatics [203] was used to produce a lot of the information in this section.

Amino acid similarity matrices (otherwise known as substitution matrices) are used by several of the methods detailed in Chapter 3. Each score in an amino acid similarity matrix tells us the frequency of mutation from one amino acid to another. Here we describe three types of substitution matrices: 1) Point Accepted Mutations (PAM), 2) Blocks Substitution Matrix (BLOSUM), and 3) the McLachlan matrix.

Point Accepted Mutation (PAM) matrices

These matrices were constructed by Dayhoff et al. [36]. These are 'based on the observed amino acid substitution frequencies in alignments of homologous protein sequences' [203, p. 82]. The sequences used to make the PAM matrices have a high sequence similarity. From the sequences a phylogenetic tree was constructed, and this was used to work out the individual mutations that had occurred between the sequences. The following ratio was then calculated:

$$\frac{\text{number of changes undergone by amino acid } a}{\text{number of occurrences of } a \text{ in the sequence set}} \tag{2.1}$$

From the ratios, the probability that each amino acid mutates into another over some period of evolutionary time is calculated. The PAM matrix shows the log value of these probabilities.

There are different versions of the PAM matrix, e.g. PAM250 is the PAM matrix for a quantity of 250 PAM units (PAM stands for Point Accepted Mutations; an accepted mutation is a mutation in the nucleotide sequence which is preserved and can be passed on to future generations.) 250 PAM means that '250 mutations have been fixed on average per 100 residues; that is, many residues have been subject to more than one mutation' [203, p. 84].

BLOSUM matrices

BLOSUM matrices, BLOSUM standing for Blocks Substitution Matrix, also depend on substitutions found in real data. These were constructed by Henikoff and Henikoff [89]. The equation used to calculate values in the set of BLOSUM matrices is:

$$s(a,b) = \frac{1}{\lambda} log \frac{p_{ab}}{f_a f_b} \tag{2.2}$$

[47, pp. 1035] a and b are two amino acids (so we complete this calculation for each possible pair), p_{ab} is the probability that we see a and b aligned in real alignments, f_a and f_b are the frequency that we see amino acids a and b in real alignments. Henikoff and Henikoff used their own BLOCKS database to produce the frequencies used in the formula. λ is a value used to ensure reasonable (whole) numbers in the matrix. The different "versions" of this matrix were produced by using different sets of pairwise alignments. So, BLOSUM62 uses a threshold of 62% sequence identity for the pairwise alignments, for example.

McLachlan matrix

McLachlan's matrix [120], like the PAM and BLOSUM series of matrices, is constructed by looking at the substitutions between amino acids in real data [182]. Unlike the PAM and BLOSUM matrices however, McLachlan's matrix uses integer scores [182]. Note: it has not been possible to find information about this matrix from the original source, as it is behind a "paywall". Accordingly, a secondary source (Tomii and Kanehisa [182]) was used for this brief description.

Having discussed concepts relating to mutations, we move onto another topic. We have alluded to the concept of species in Section 2.1; this is described in more detail in the next section.

2.3 Ancestry and phylogenetic trees

As mentioned earlier in this chapter, examples of species include humans, domestic cats, and domestic dogs. All species that exist, and have ever existed, are related by the so-called "tree of life". We know that intuitively, a fox is "more related" to a dog than it is to a human, but here we formally define the concept of a species. 'A biological species is a group of organisms that can reproduce with one another in nature and produce fertile offspring. Species are characterized by the fact that they are reproductively isolated from other groups, which means that the organisms in one species are incapable of reproducing with organisms in another species. ... Species can also be defined based on a shared evolutionary history and ancestry' [127]. This last statement should make sense within the context that foxes are more related to dogs than humans. It is worth noting that a caveat of the definition quoted is that it only really applies to organisms that reproduce sexually; the last phrase in the quote (about definition of a species based on shared ancestry) is more applicable generally.

But how do new species come about? There are several models [162]. One of these is 'ecological speciation', which is how speciation might arise from different populations of a species adapting to different ecological conditions (environments) such that they become two separate species. Another involves sexual selection. An example of this is how species from the Peromyscus (deer mice) genus can produce fertile hybrids under lab conditions, 'but rarely produce hybrids in the wild, even when sympatric' [179] (sympatric meaning they live in similar areas geographically).

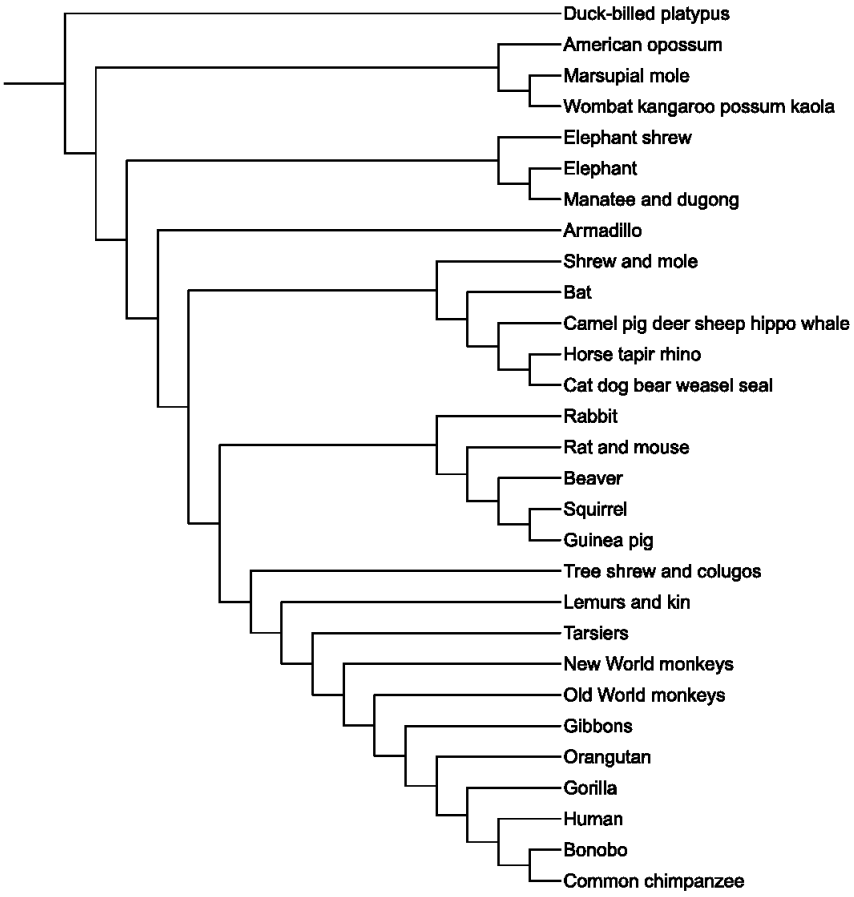

Figure 2.7: A tree showing how an array of species are related through evolution. Image is available under the GNU Free Documentation License [90].

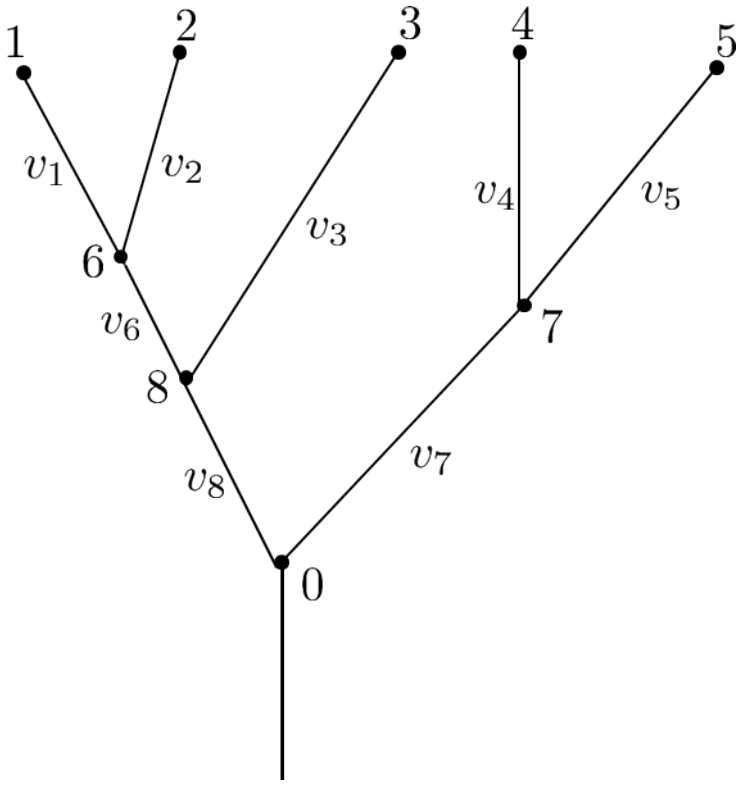

Figure 2.8: An example of a phylogenetic tree.

Species are related to each other through the process of evolution. For example, Figure 2.7 shows how a selection of species are related; it shows that the bonobo and common chimpanzee shared a common ancestor (that is, a common ancestral species, which no longer exists, is the parent ancestor of two different species, the bonobo and the common chimpanzee. These two species came about through a speciation event, such as we have described above). The structure given in this figure is a phylogenetic tree, or phylogeny. We describe phylogenies now in a little more detail.

2.3.1 Phylogenetic trees

A phylogenetic tree shows us how extant sequences – sequences that currently exist – evolved from their ancestors (see Figure 2.8 for an example). The lines are called *branches*, and the small circles at the intersections or ends of branches are called *nodes*. The nodes at the very top of the tree (numbered 1-5) are the *tips*, or *leaves* of the tree, and each represents an extant sequence (in our case, we are working with protein sequences). Trees can be drawn with either the leaves at the top or the bottom. The nodes at the intersections of branches are *internal nodes*. At each point in the tree where an internal node splits off into two branches some kind of divergence occurred between sequences; for example, at node 8 two different sequences emerged. A *clade* is a sub-tree within a whole tree, and it includes a particular node and

its descendants.

2.3.2 Homology

Now that we know how species can be related, we introduce the concept of homology, and of homologous protein sequences, which can be categorised as paralogous or orthologous. Two sequences are paralogues if they are within the same species (as part of the same genome); that is, they occurred when one gene duplicated within a genome (these two genes can then go on and accumulate mutations, making them different). Such a duplication could happen during the process of meiosis, which is the cell division involved in the production of eggs and sperm. During this process, there is a recombination event, in which genes are swapped between homologous chromosomes (chromosomes in a pair, one from the mother and the other from the father). If the chromosomes are not lined up exactly, this can cause gene duplication.

On the other hand, two sequences are orthologues if they occur within different species, so they appear when a speciation event occurs. One ancestor species produces two descendant species, each of which has its own copy of a gene [109].

Now that we have discussed the ideas behind ancestry and relationships between species, we now discuss how these relationships can be represented.

2.4 Phylogeny representation and construction

First we describe the Newick format, a text-based approach to describing a phylogeny, and then discuss a method for reconstructing phylogenies.

2.4.1 Newick format

This is a format for the representation of trees; a set of brackets $()$ encapsulates the children of the node represented by those brackets, and commas $,$ separate the children of nodes. Nodes can be given labels, and branch lengths are represented by a colon and then a number, either after a node name or after a closing bracket, if that (internal) node has no name. So, for example, $((A : 2.5, B : 0.5) : D : 1.0, C : 2.5)$; represents the tree in Figure 2.9, created using the web application Phylodendron [72] (we added the branch lengths).

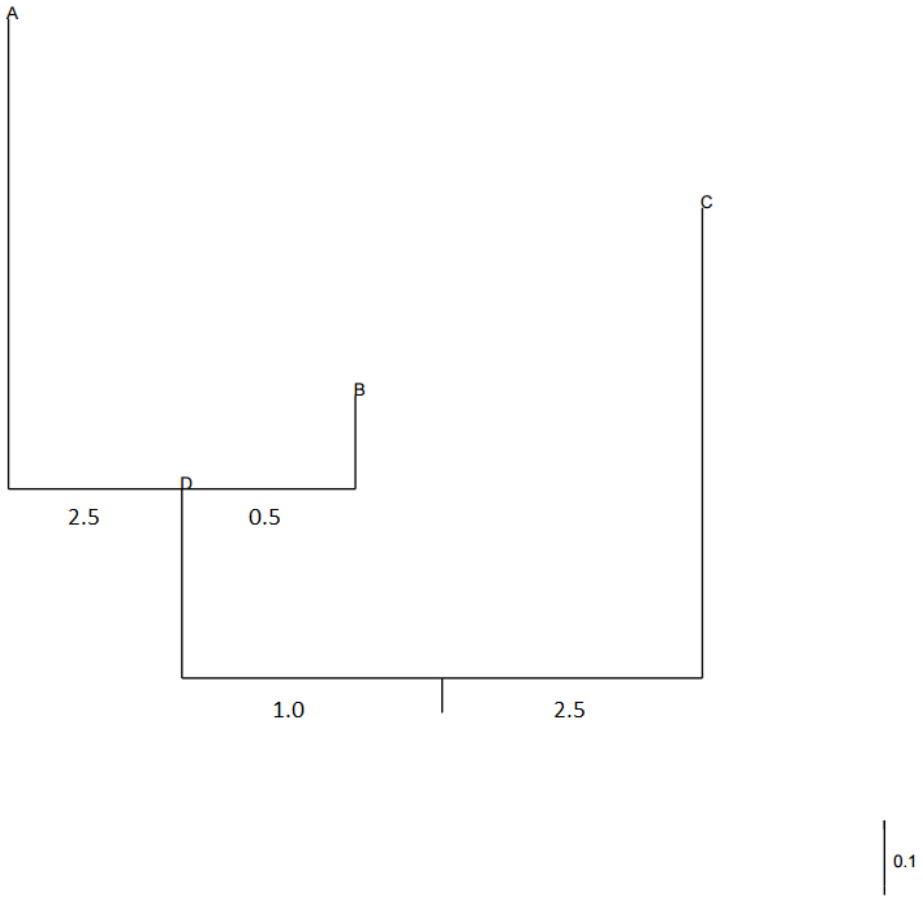

Figure 2.9: An example tree, represented by $((A : 2.5, B : 0.5) : D : 1.0, C : 2.5)$; in Newick format, created using the web application Phylodendron [72].

2.4.2 ClustalW2

ClustalW2 [107] is an alignment program (a program for building multiple sequence alignments, which we define in Section 2.6) built upon the earlier alignment program Clustal W [180]. This program has the option to find the tree for a given alignment using the Unweighted Pair-Group Method with Arithmetic mean (UPGMA) algorithm.

UPGMA [48] is a clustering algorithm; it works from the leaves up to find where the branching events occur on our constructed phylogeny. It works by first constructing a distance matrix between each pair of species, and then it uses this matrix to construct the phylogeny. "Unweighted" means that all of the distances in the matrix contribute equally to calculations, "pair-group" means pairs of sequences (or pairs consisting of a single sequence with an already-constructed group, or pairs of groups) are clustered, and "arithmetic mean" means that the distances are calculated using the arithmetic mean. So if, for example, we wish to calculate the distance between the group (A, B, C) and the group (D, E), where A, B, C, D, and E are sequences (leaves in our constructed tree) then we calculate the mean of the distances between (A and D), (A and E), (B and D), (B and E), (C and D), and (C and E).

As mentioned previously, at each node in a tree there is a sequence. We only know for certain the sequences at the leaves of a tree (as these are from extant, or currently existing, species), but for some of the methods for detecting molecular co-evolution we need to predict the ancestral sequences (those at the internal nodes). To do so, we must use ancestral sequence reconstruction.

2.5 Ancestral sequence reconstruction

Given a set of (nucleotide or amino acid) sequences from a set of extant species, ancestral sequence reconstruction aims to find the most probable sequences of the ancestors to those species, e.g. for the tree shown in Figure 2.10, we know what the characters at certain positions of a protein are for the species at nodes 4–7, but we want to find out what would probably have existed at those positions for the species at the nodes 1–3.

2.5.1 Fitch's algorithm

Fitch's parsimony algorithm [59] is one method which allows us to reconstruct ancestral sequences. It assigns states (amino acid characters) to a tree

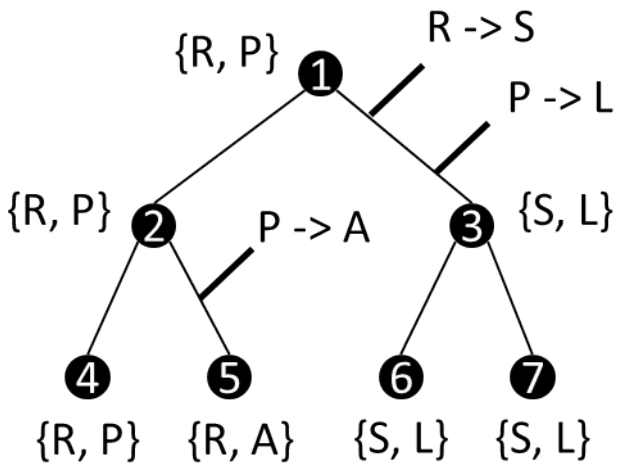

Figure 2.10: An example of how co-evolution may occur – see the main text for explanation.

topology (given as input) so that the minimal number of mutations evolve the given sequence data. This means that it is a Maximum Parsimony method. This method assumes that we know the ancestral relationships between the given sequences (the associated phylogenetic tree). We also assume that we are dealing with orthologous sequences (from different species) and that we are looking at a Markov Chain model of evolution, so that the probability of a character mutating from one amino acid to another depends only on the current state, and not on any previous states.

It should be noted that Fitch's algorithm is intended for use with nucleotide sequences, and naively mapping this to use amino acid (protein) sequences instead would imply that any amino acid can change to any other with equal probability, which is obviously not true (although we could change the model to take into account the likelihood of "mutating" from one amino acid to another).

2.5.2 Maximum Likelihood

Maximum Likelihood (ML) is another method for ancestral sequence reconstruction; it aims to work out the character at each of the nodes such that the probability of the characters at the leaf nodes is maximised, given a tree topology as input [98]. A likelihood calculation is carried out at each node of the tree, iterating over all of the possible characters. We can look at each node in turn to find the marginal likelihood, or at all of the nodes at once, to find the joint likelihood. The joint likelihood is more computationally expensive to calculate than the marginal likelihood, but is generally accepted to be more accurate.

Sequence number \ Index into the sequence	0	1	2	3	4	5
0	G	G	C	G	C	C
1	C	A	C	T	T	T
2	A	G	C	T	G	C
3	T	C	C	A	A	T

Figure 2.11: A toy MSA, where each of the characters is a nucleotide, according to the alphabet mentioned previously.

We now describe the concept of multiple sequence alignments, which are used as input for all of the methods we consider for detecting molecular co-evolution.

2.6 Multiple Sequence Alignments

A multiple sequence alignment is the result of taking multiple (nucleotide or protein) sequences from extant species and aligning them so that we can visualise the similarities between them. The co-evolution detection methods that shall be encountered within this thesis use protein sequences. An example alignment is shown in Figure 2.11. There are two main categories of alignment algorithm; those that produce global alignments, and those that produce local alignments. A global alignment is one which "[spans] all the nucleotides or amino acids in the sequences that have been submitted for alignment" [203, pp. 739]. A local alignment is "an alignment of the most similar regions of a nucleotide or amino acid sequence ignoring other segments of the sequences" [203, pp. 741], i.e. it is only for a small segment of the sequences, and does not require every character of every sequence to be aligned. So, the fundamental difference between the two is whether they align the entirety of the sequences, or only a subset of the characters in the sequences. A selection of different alignment programs are described in Section 4.2.1, along with our justification for why we chose to use the one we did.

In an alignment, each row's values come from a different sequence, and the general theory is that through aligning the sequences each column will tell us how those sequences evolved – each character is inherited from an ancestral character [112]. For example, if we had the nucleotide alignment given in Figure 2.11, let it be that the four sequences in this alignment evolved according to the tree structure shown in Figure 2.12, then we could say that

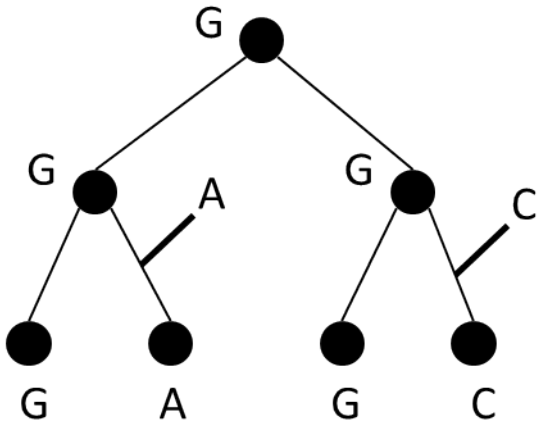

Figure 2.12: According to column 1 in Figure 2.11, a tree showing how the characters at the leaf nodes (from that column of the alignment) may have evolved from a common ancestral character. The lines pointing to different branches symbolise where a mutation occurred.

sequence 0 corresponds to the left-most leaf of the tree, sequence 1 to the leaf directly to the right of this, and so on. We could then say that the sequences that the ancestors possessed could have had the characters given at the internal nodes at position 1 of their respective sequences.

We can also have gaps in our alignment (represented by the dash - character); these symbolise that either a character was deleted at that position in that sequence, or it could mean that a character was inserted in another sequence at that column.

There are several ways of formatting alignments. FASTA and PHYLIP, which are commonly used, are described here.

2.6.1 FASTA

In a FASTA alignment (named after the FASTA, or Fast-All, program), the information for each sequence is given sequentially. First, there is a line starting with the $>$ character, followed by the sequence name, and then on subsequent lines is the associated sequence. For example:

```
>gi|2924948
MAAALRRGYKGLGFSFELTEQQKEFQTIARKFAREEIIPVAPDYDKSGEY
PFPLIKRAWELGLINTHIPESCGGLGLGTFDACLITEELAYGCTGVQTAI
EANSLGQMPVIIAGNDQQKKKYLGRMTEQLMMCAYCVTEPSAGSDV
>gi|1753489
------------QMSFELSDTQKEIQSHAIKFSKDVLVPNAAKFDKSGEF
```

```
PWEIVKQAHSLGFMNTIIPEKYGGPGLSNLDTALIVEALSYGCSGLQIAI
FGPSLAAAPICLSGTEEQKKKYLGMLAAEPIIASYCVTEPGAGSDV
>gi|6680618
MAAAFRRGCRGLGFSFELTEQQKEFQATARKFAREEIIPVAPEYDKSGEY
PFPLIKRAWELGLINAHIPESCGGLGLGTFDACLITEELAYGCTGVQTAI
EANSLGQMPVILAGNDQQKKKYLGRMTEQPMMCAYCVTEPSAGSDV
>gi|3901902
MTMLFNKVLRNGGFSFELTEQQKEFQEVARKFAREEIVPAAPSYDRSGEY
PFPLIKRAWELGLMNGHIPEDCGGMGLGIFDACLITEELAYGCTGVQTAI
EANSLGQMPVIIAGNDAQRKKYLGRMTEEPLMCAYCVTEPGAGSDV
```

2.6.2 PHYLIP

Another way of formatting an alignment is the PHYLIP format (named after the PHYLIP, or Phylogeny Inference Package, program). The first line contains the number of sequences and the length of the alignment. Here we describe the sequential PHYLIP format; this means that, again, information on each sequence is presented in turn. The first 10 characters of a line will be the sequence name; the rest of that line will be the actual sequence; any lines after this one will be the rest of the sequence, if necessary.

```
 4 146
gi|2924948MAAALRRGYKGLGFSFELTEQQKEFQTIARKFAREEIIPV
APDYDKSGEYPFPLIKRAWELGLINTHIPESCGGLGLGTFDACLITEELA
YGCTGVQTAIEANSLGQMPVIIAGNDQQKKYLGRMTEQLMMCAYCVTEP
SAGSDV
gi|1753489-----------QMSFELSDTQKEIQSHAIKFSKDVLVPN
AAKFDKSGEFPWEIVKQAHSLGFMNTIIPEKYGGPGLSNLDTALIVEALS
YGCSGLQIAIFGPSLAAAPICLSGTEEQKKKYLGMLAAEPIIASYCVTEP
GAGSDV
gi|6680618MAAAFRRGCRGLGFSFELTEQQKEFQATARKFAREEIIPV
APEYDKSGEYPFPLIKRAWELGLINAHIPESCGGLGLGTFDACLITEELA
YGCTGVQTAIEANSLGQMPVILAGNDQQKKYLGRMTEQPMMCAYCVTEP
SAGSDV
gi|3901902MTMLFNKVLRNGGFSFELTEQQKEFQEVARKFAREEIVPA
APSYDRSGEYPFPLIKRAWELGLMNGHIPEDCGGMGLGIFDACLITEELA
YGCTGVQTAIEANSLGQMPVIIAGNDAQRKKYLGRMTEEPLMCAYCVTEP
GAGSDV
```

We now describe co-evolution, and how it appears. Methods for detecting co-evolution will be discussed in the next chapter.

2.7 Co-evolution

The first use of the term "co-evolution", according to Pazos and Valencia [147], was by Ehrlich and Raven in 1964 [49], who did a study on the relationships between different families of butterflies and the plants they feed on. This refers to co-evolution on a species level. In contrast, *molecular co-evolution* describes such relationships on a genomic, or proteomic, level.

Molecular co-evolution (or correlated evolution) is where we have 'the direct physical effect of one sequence position on the other' [177, pp. 341]. Mutations occur randomly, but they can be fixed in a population through the process of natural selection. For example, if there were a mutation at one site which somehow destabilised that protein's structure or stopped the protein from functioning properly, it could be compensated for by another mutation at a different site (or multiple mutations individually at different sites), in order to "neutralise" the first mutation. We should note here that mutations occur randomly, and can be kept or lost depending on the effect they have on the organism's fitness. If a mutation has a positive effect on the organism's fitness, it will be more likely to be kept, but if it has a negative effect, then it will be more likely to be lost [203].

Another example of co-evolution is where we have multiple sites which "evolve together" (i.e. at which mutations are fixed) in order to alter some aspect of the protein in a way which changes that protein's function to allow the organism to better adapt to its environment.

Molecular co-evolution has been detected in previous studies in order to improve the quality of protein structure predictions, for example. Reviews of these are given by Caporaso et al. [22], Codoñer et al. [31], and Ng and Henikoff [133]. There are also plenty of individual papers which implement various methods for detecting molecular co-evolution (a non-exhaustive list is [1, 16, 17, 20, 24, 25, 28, 29, 30, 32, 40, 42, 44, 43, 45, 46, 56, 58, 65, 68, 69, 73, 74, 75, 76, 78, 84, 86, 95, 101, 104, 110, 114, 116, 121, 131, 134, 138, 141, 146, 147, 152, 153, 154, 156, 168, 175, 178, 181, 184, 189, 191, 193, 195]).

The co-evolution "signal" between two positions in a protein, denoted i and j, can be divided into different parts (this assumes an additive model) [17, p. 1627]:

$$`C_{ij} = C_{phylogeny} + C_{structure} + C_{function} + C_{interactions} + C_{stochastic}' \quad (2.3)$$

What this equation means is that we can detect a pair of positions as co-evolving due to the various components in the equation (each component contributes to the overall signal). However, we are not equally interested in each of these components. The components of the equation are defined:

$C_{structure}$ One substitution is compensated for by another in order to maintain the protein's structure (see the example earlier in this chapter of the substitution of a "large" amino acid for a smaller one) or to evolve new structures – those positions co-evolving due to structure will be relatively close in the 3D structure [24], although they may not be close in the amino acid string (the primary structure)

$C_{function}$ One substitution is compensated for by another in order to maintain the protein's function or to evolve new functions

$C_{phylogeny}$ For orthologous genes (those which occur in different, related, species), for example, mutations which are fixed in the sequences of ancestral species can be propagated down to extant (currently existing) species. This can mean that co-evolution can appear to be occurring between sites when it is really deriving from the "relatedness" of the sequences

$C_{interactions}$ The three sources of co-evolution described above are not statistically independent; $C_{interactions}$ accounts for this (this component can also be related to interaction between different proteins [31])

$C_{stochastic}$ Co-evolution caused by chance (and anything else not covered in the above types of co-evolution)

C_{ij} This is the co-evolution signal that we can observe between positions i and j in a protein. When we look at this signal, we want to remove certain types of co-evolution described above (e.g. $C_{phylogeny}$) to get to the "truly" co-evolving sites.

It should be noted that molecular co-evolution is a biological process which we cannot directly observe; to find it we must consider the concept of covariation; this is where "some pairs of residues co-occur in multiple sequence alignments more frequently than expected" [176, pp. 2456].

It is assumed that molecular co-evolution (which we cannot currently directly observe) causes covariation (which we can observe). We shall now give an example of how we plan to use covariation to detect co-evolution; see Figure 2.10. Imagine we have four extant (currently existing) species, shown in the figure as 4, 5, 6, and 7, and that these are related according to

the tree shown (1, 2, 3 are ancestral species). (We are only concerned with the same two positions in each protein string, and the characters that appear at these positions, for the sake of this example.) Further imagine we can look back in time and "see" all the mutations that occurred to a particular protein within each species, as the ancestral species evolved over time into those we have today. In the figure, the characters in curly braces represent two different characters in that species' protein string for the protein we are focusing on. When a line points out from a branch between two species (nodes on the graph), it shows where mutations become fixed at those positions. When we are looking for molecular co-evolution, we are hoping to find such events as can be seen on the branch from 1 to 3 – where a mutation becomes fixed in one position, and then (in "response") a mutation becomes fixed at another position – this is what we think is causing covariation in the ensuing multiple sequence alignment. In the rest of this thesis we use the terms "molecular co-evolution" and "covariation" interchangeably.

2.7.1 Sources of randomness in correlation signal

As mentioned above, there is correlation between all pairs of sites within a protein due to chance. The amount of this correlation depends on a number of variables:

- Number of sequences in the alignment
- Number of species in the alignment (there can be multiple sequences from a single species in an alignment), and how "related" these species are (e.g. in Figure 2.10, we could say species 4 and 5 and more related than 4 and 6)
- Alignment length – the number of random correlations increases as the length increases
- Sequence conservation – the more conserved, the lower the number of random correlations. If a position in an alignment is fully conserved, it only contains one type of amino acid (one character only). Sequence conservation depends on:
  - the presence of functional domains, which 'are distinct functional and/or structural units in a protein' [163, para. 1] and are conserved; internal interactions between amino acids within a protein; or the protein having a hydrophobic core (the amino acids which make up the core tend to be more conserved than those not part of the core) [113]

- the actual amino acids in a sequence – for example, the amino acid methionine can only be coded for by the codon (group of three characters) AUG, meaning a mutation to any of the bases is going to cause a change to the corresponding amino acid. In contrast, proline is coded for by four different codons (CCA, CCC, CCG, CCT), meaning any change to the final nucleotide in the codon will mean that the amino acid does not change. This means that during the evolutionary process, prolines are more likely to be retained than methionines in the protein sequence, increasing the likelihood of this position being more conserved
- other factors which change the likelihood of particular bases to mutate, e.g. local GC content (the percentage of bases which are either G or C), the proximity and degree of genetic linkage[1] to other genes that are highly conserved

Co-evolving sites are thought to be important to a protein's structure or function, and we can find these by looking at multiple sequence alignments. Another measure we can use from multiple sequence alignments to find important sites is conservation, which we describe now.

2.8 Site conservation

Taking the example in Figure 2.11, we say that column 2 is highly conserved as it has little (or in this case, no) variation across the sequences. If a site is highly conserved across sequences, then it is under strong negative selection, and so that site is likely to be important to that particular protein's function. From this we can say that mutations at that site would probably be hazardous to the protein's function (if an amino acid changes to another with very different chemical properties, for example).

Current bioinformatics algorithms take conservation into account when classifying mutations at a site according to their likely pathogenicity [105].

2.9 Parallel computing

This project will involve attempting to improve the runtime performance of methods for detecting molecular co-evolution such that they can be applied to much larger data sets than has been done previously (the human proteome,

[1] 'Linked genes sit close together on a chromosome, making them likely to be inherited together' [188, para. 1]

or all human proteins). The first step towards this goal is to decide which approach to take in doing so, and which parallel architecture type to use. We begin by reviewing a common classification for parallel architectures, Flynn's taxonomy, before outlining our reasons for choosing the architecture we did (NVIDIA graphics cards) and then describing this particular architecture in more detail.

Flynn's taxonomy [64] is a well-used classification system for parallel architectures. It is defined in terms of instruction streams and data streams. The instruction stream is the series of instructions completed by the system, and the data stream is the data that the system performs its instructions on. There are four categories:

- Single Instruction, Single Data (SISD): the typical serial computer, it performs one instruction on one piece of data, then once that has completed (and only then), it can perform another instruction on another piece of data, and so on
- Single Instruction, Multiple Data (SIMD): the same instruction is performed on lots of data at once, before moving onto the next instruction (with another lot of data). This describes CUDA's architecture, as well as processor array architectures. 'A processor array is a vector computer implemented as a sequential computer connected to a set of identical, synchronized processing elements capable of simultaneously performing the same operation on different data.' [157, pp. 62]
- Multiple Instruction, Single Data (MISD): different instructions are performed on the same piece of data at the same time; this is rarely used in real systems
- Multiple Instruction, Multiple Data (MIMD): different instructions are performed on different pieces of data; examples of this kind of system are massively parallel processors (MPPs) (a type of system whereby lots of microprocessors are connected together and work in parallel), and clusters [118]

Of the different options available (supercomputers, clusters, graphics cards) we chose to use graphics cards, for the reasons that they are relatively cheap (in Chapter 7 we look at the costs of using GPUs versus supercomputers in more detail), readily available to the average researcher (not all researchers may have access to a supercomputer or a cluster), and also in the case of NVIDIA GPUs particularly, they are easy to use if one knows how to program in C. We shall now introduce GPUs, and the NVIDIA architecture.

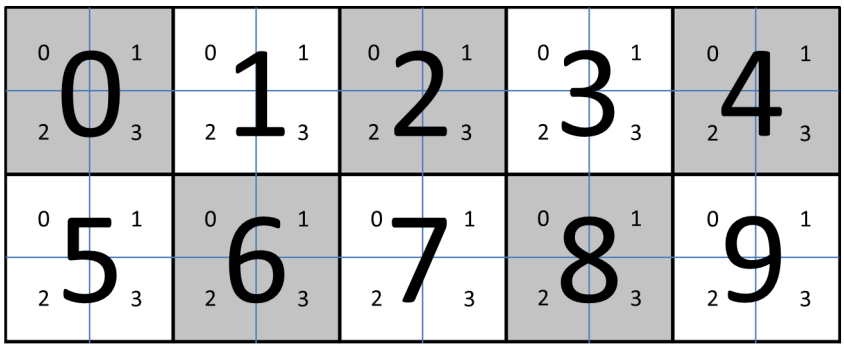

Figure 2.13: An example of how a grid is divided into blocks, and blocks into threads. The entire rectangle is the grid, and each large square (number with large font and alternately shaded and unshaded) is a block. There are 10 blocks, numbered 0–9, and these are arranged such that CUDA's `gridDim.x` variable is equal to 5 (the number of blocks in the x dimension), and the `gridDim.y` is equal to 2 (the number of blocks in the y dimension). Each block has 4 threads such that `blockDim.x` and `blockDim.y` both equal 2.

Graphics Processing Units, or GPUs, generally refer to the processing units in graphics cards, although the terms "GPU" and "graphics card" do tend to be used interchangeably, as we do here. They have always been commonly used for rendering graphics, but in the year 2003 the term GPGPU (General Purpose GPU) was coined [88] to represent the growing number of cases where people were using GPUs to do more general computation on large amounts of data.

NVIDIA is one designer of such generally programmable GPUs; the architecture of these graphics cards fits into the SIMD (Single Instruction Multiple Data) class in Flynn's taxonomy [64], which means that one instruction is carried out on large amounts of data, before moving on to the next instruction.

NVIDIA graphics cards can be programmed using the programming language CUDA (Compute Unified Device Architecture), which is like the language C, but with some extensions for transferring data to and from the GPU, and for executing programs (called *kernels*) on the GPU.

Work on the GPU is organised into grids, and each grid is composed of blocks. Each block has threads, and each thread works on the code contained within a kernel. (See Figure 2.13 for an example showing how a grid of blocks of threads could be organised.) Each launch of a kernel (program) can specify a different division of work into blocks, depending on the application.

For example, say we had the kernel `incrementValues` shown in Listing 2.1. This is a very simple kernel which takes an integer array `in` and outputs another integer array, called `out`. Imagine our `in` array consists of 10 integers only (such an example would almost invariably be more efficient to run on the CPU as normal, but we use this very trivial example for what it is, an example). We decide we want to launch 2 blocks of threads, each block having 5 threads, giving us 10 threads total (1 per element in the input array). Each thread executes this kernel independently of the others. When the thread gets to line 3, it substitutes into the equation its own values of `blockIdx.x` and `threadIdx.x`. For example, the third thread in the second block would have `blockIdx.x` equal to 1 (counting from 0) and `threadIdx.x` equal to 2.[2] The `blockDim.x` is, in this case, 5 (5 threads per block in the x dimension), and so the `idx` value for this thread equals 7. Line 4 reads in the seventh value from the `in` array, adds 1 to it, and then outputs this into the `out` array.

```
1 __global__ void incrementValues(int in[], int out
    []) {
2
3         int idx = (blockIdx.x * blockDim.x) +
    threadIdx.x;
4         out[idx] = in[idx] + 1;
5
6 }
```

Listing 2.1: An example CUDA kernel.

There is not only a hierarchy of threads in the CUDA architecture; there is also a memory hierarchy. Each thread has its own local memory. Each thread block has its own shared memory, accessible to all the threads in that block. There is also global memory accessible to all threads in all blocks. The "more local" the type of memory, the better the performance in reading from and writing to that memory, so the thread-local memory has the best performance, while global memory has the worst.

To the best of our knowledge, only one paper has previously been published on a method for identifying co-evolving sites using CUDA; this is the paper by Waechter et al. [191]. Although this is a good place to begin, we did not want to use this implementation directly due to it being an implementation of

[2]The reason for the `x` at the end of `blockIdx.x` and `threadIdx.x` is that blocks can be three-dimensional, and so can grids (here we have only a one-dimensional grid of blocks, and one-dimensional blocks of threads).

Mutual Information, which is one of the more naive methods, and there are other methods which can give better performance in detecting co-evolution (we review the different categories of method in the next chapter).

2.10 Complex brain disorders

Later in this thesis, we shall evaluate the results of the proteome-wide analysis of co-evolving sites in the context of mutations associated with complex brain disorders. The disorders we shall be focusing on are schizophrenia, autism spectrum disorder, and intellectual disability (the Fromer et al. [66] dataset, which we shall be using, is from individuals with these disorders).

The types of mutations contributing to schizophrenia and autism include both common single nucleotide polymorphisms (SNPs) [55] and rare mutations: single nucleotide variants (SNVs) and small insertions and deletions (indels) [66], and copy number variants (CNVs), which are large insertions or deletions to the genome [102, 151, 139].

There has been shown to be genetic overlap between psychiatric disorders, in particular there are CNVs that can influence risk for schizophrenia, autism, and intellectual disability, as well as developmental delay and attention deficit hyperactivity disorder (ADHD) [139]. In addition to common genetic influences to these disorders, these disorders all tend to cause cognitive impairments [142]. There is a significant amount of co-morbidity (having two disorders at once) between schizophrenia and intellectual disability, and between schizophrenia and autism. These disorders can all be placed on a gradient of cognitive impairment, ordered from most severe to least: intellectual disability, autism, ADHD, schizophrenia, bipolar disorder [142].

There is a lot of difficulty in working out which small, rare mutations (SNVs and indels) contribute to disease. This is because there is not a large increased burden of SNVs/indels in individuals with these disorders (so many of the mutations they have may not contribute to their disorder). Also, the effect sizes of these mutations are relatively modest [170], so we need a large sample size to reliably identify mutations contributing to disease. We can try and identify which mutations are more likely to be functionally damaging to the gene/protein (and hence more likely to contribute to disease). Earlier in this chapter we introduced the concept of conservation; this can be used to infer the positions at which mutations may be damaging to protein function. In addition, there are programs, such as SIFT [132] and PolyPhen [160] (discussed in more detail in Chapter 8) which can be used to classify mutations

as damaging or not. There is evidence that mutations that are predicted to be damaging do contribute to disease [70, 171].

2.11 Summary

In this chapter we introduced first some basic biological concepts, including what DNA, RNA, and proteins are and how they are made. We linked this to the concept of ancestry, and discussed the concept of multiple sequence alignments as the input we shall use in our methods for detecting co-evolution, which we defined. Site conservation as another measure for finding sites important to a protein's function was described. We then introduced GPUs as the approach to improving the performance of these detection methods. The next chapter is an overview of the literature surrounding molecular co-evolution detection methods. We shall also choose which methods we wish to focus on.

Chapter 3

Literature Review

This chapter provides an overview of the methods that are currently in use for identifying molecular co-evolution, and is divided into sections based on the general approach taken in each method. This classification was influenced by that of Codoñer and Fares [31].

All of these methods use MSAs (Multiple Sequence Alignments) as input; some also use other types of data; outputs are usually some kind of co-evolution score telling us, for each pair of columns in the alignment, how likely that pair is to be co-evolving.

All of the methods for detecting molecular co-evolution can be abstracted into a general template, given below:

1. Find raw pairwise signal
2. Account for stochasticity (or randomness) ($C_{stochastic}$)
3. Account for sequence similarity/relatedness (phylogenetic signal – $C_{phylogeny}$)
4. Find "clusters" of co-evolving sites

The two main types of method are non-parametric and parametric methods. Non-parametric methods are different to parametric methods in that they do not use information from a phylogenetic tree associated with the input MSA. Parametric methods use information on the phylogenetic tree associated with the MSA, and generally require that we make some assumptions about parameters such as the rate of evolution of positions, for example.

All of the methods we describe are performed on each pair of columns in an alignment under the assumption that the result we obtain for columns i and j is the same as for the pair of columns j and i. This means that for n columns, we compute $(n * (n-1))/2$ comparisons.

3.1 Non-parametric methods

In this section we shall introduce categories of methods which do not use phylogenetic tree information when detecting molecular co-evolution. These are Mutual Information (MI), Observed Minus Expected Squared (OMES), perturbation-based algorithms, McBASC (McLachlan-Based Substitution Correlation), PSICOV (Protein Sparse Inverse Covariance) and DCA (Direct Coupling Analysis), and Sequence Divergence-based Approximation. For each of these categories of methods, we describe the basic idea behind that method, as well as the advantages and disadvantages of that method from a theoretical point of view in terms of how well it takes biological information into account. Once we have considered each category individually, we shall compare these methods according to studies already published.

3.1.1 Mutual Information (MI)

This category is arguably one of the most well-used types of method in detecting co-evolving pairs. The basic method is taken from the Information Theory defined by Shannon [167]. The Mutual Information (MI) equation is given as:

$$MI_{ij} := \sum_{a,b \in A} p_{ij}(a,b) log_2 \left(\frac{p_{ij}(a,b)}{p_i(a) p_j(b)} \right) \tag{3.1}$$

The equation as seen above is from a study by Waechter et al. [191, p. 50]. i and j are columns in the alignment, A is the alphabet of characters (in our case, amino acids), and a and b are individual amino acids. $p_i(a)$ and $p_j(b)$ are the probabilities of observing character a at position i and character b at position j, respectively, and $p_{ij}(a,b)$ is the probability of observing character a at position i, and at the same time observing character b at position j. Other studies which used this approach are: [103, 9, 10, 116, 73, 24].

Changing the idea of MI

Ackerman et al. [1] developed a class of methods, called "differential binary methods", which use the original MI equation, but first translate the input

alignment into a binary matrix by putting, in each row, a 1 if there is a difference between the character present at the corresponding position in the MSA, and the character present in the same column in the row above (and a 0 if the characters are the same) – the first row of the matrix is set to all 0s. The ordering of the sequences in the original MSA influences the resulting binary MSA; these methods optimise the binary MSA such that there are the fewest possible 1s. This gives us lower values of MI calculated from the binary matrix, and exposes the motivation for developing these methods: 'The overall decrease in the average MI of the alignment allows for an easier identification of concerted amino acid changes in sequences throughout the alignment by reducing the entropy associated noise' [1, para. 38]. The Mutual Information matrix calculated from this binary matrix can be further processed according to Gloor et al. [74], which gives us what is called the ZPX matrix. We square this to get the ZPX2 matrix, and this result is named by Ackerman et al. the dbZPX2 matrix (Differential Binary ZPX2; the version of the alignment which is represented in binary, described above, is called the differential binary matrix). There are also two methods described by Ackerman et al. which include information on the specific amino acids in the alignment (and not just whether there is a difference between the character in this row and the character immediately above); these are called dgbZPX2 and nbZPX2.

Burger and van Nimwegen [18] derived a statistic which has similarities to MI, called logR. Their method actually includes phylogenetic information (and is therefore a parametric method). They also incorporate the APC (Average Product Correction) (described below) to remove the influence of spurious correlations that may appear because the sequences in the alignment are from evolutionarily related species.

Normalisations

As mentioned in Section 2.7, the overall co-evolution "signal" that comes from two alignment columns can be divided into components, some of which we are not interested in (those components from the fact that the sequences used in the alignment are related ($C_{phylogeny}$) as well as co-evolution signal appearing by chance ($C_{stochastic}$)). Because we are not interested in these components, which are essentially noise, we need to do our best to eliminate them. One way of doing so is to "correct" the test statistic, through normalisation, for example. We now look at a few methods where the authors have attempted to account for these background components through normalisation.

Martin et al. [116] looked at different MI-specific methods of normalisation, which involve dividing the MI equation by values involving entropy, etc. (The MI equation can be defined in terms of the entropies of alignment columns.) Merkl et al. [121], in their H2r method, used a U statistic (referring to Press et al. [155]) which normalises MI by the summation of the individual entropies for the two columns we are looking at. Dunn et al. [42] also described two normalisations of MI – the Average Product Correction (APC), and the Average Sum Correction (ASC). Tillier and Lui [181] introduced the entropy-weighted dependency ratio, which is another way of normalising MI.

Adding biological information

Gao et al. [69] took the Mutual Information equation and added an amino acid background distribution (MIB'), and also information about amino acid physicochemical properties (MIP'). Using information on the background distribution allows us to remove the influence of correlations caused by the sequences being evolutionarily related. Using an amino acid's physicochemical properties would take into account how often amino acids from different physicochemical "groups" appear together at different positions and would allow us to work out when it is "acceptable" to replace an amino acid in one group with another amino acid from a different group. In addition, for part of their study, Atchley et al. [10] recoded the amino acids in the alignment by whether they are hydrophobic, hydrophilic, or neutral, to see if co-evolving pairs tended to have certain characteristics. They also looked for correlations in terms of amino acid size.

Advantages

- Normalisations can be used to remove some of the background noise
- The original MI equation as seen in Equation 3.1 is easy to use and understand; because it does not use biological information it could be used as a sort of benchmark for the rest of the results to be compared against

Disadvantages

- If we simply use the MI equation as seen in Equation 3.1, then we are making no distinction between, for example, substituting D (a negatively charged amino acid) for E (another negatively charged amino acid) and substituting D for K (a positively charged amino acid)
- (Specific to the methods of Gao et al. [69] and Atchley et al. [10]) we would have to make a choice as to how we would categorise amino

acids according to physicochemical properties (the wrong assumptions could give us an incorrect result)

3.1.2 Observed Minus Expected Squared (OMES)

This method of identifying co-evolving pairs was described by Kass and Horovitz [99] and is basically equal to a χ^2 test:

$$\chi^2(i,j) = \sum_n (N_{n,OBS} - N_{n,EX})^2 / N_{n,EX} \tag{3.2}$$

i and j are the two columns we are focusing on, n is equal to the number of possible different pairs of amino acids there are in columns i and j (and each value of n is equal to a different pair of amino acids X and Y, where X is any of the amino acids which appear in column i, and Y is any of the amino acids which appear in column j), $N_{n,OBS}$ is the actual number of times we see the pair X and Y at positions i and j, respectively, and $N_{n,EX}$ is the expected number of times we see the pair X and Y at positions i and j, respectively. This is equal to $Nf_{X,i}f_{Y,j}$ where N is the number of rows in the alignment, $f_{X,i}$ is the frequency of X at position i, and $f_{Y,j}$ is the frequency of Y at position j. Noivirt et al. [135] also used the basic OMES equation for their study.

A variation of this method was proposed by Fodor and Aldrich [65]. The main equation was redefined as:

$$\sum_l^L \frac{(N_{obs} - N_{ex})^2}{N_{valid}} \tag{3.3}$$

so the bottom portion of the equation has been replaced with N_{valid}, which is the number of rows in the alignment, subtracting the number of rows with a gap at either (or both) position(s). This equation was also used by Kowarsch et al. [104].

Advantages

- This method is easy to understand, like Mutual Information, and is fairly simple to compute

Disadvantages

- Like Mutual Information, this class of method does not consider the actual meaning of the amino acids in the alignment, only viewing each amino acid as a different character

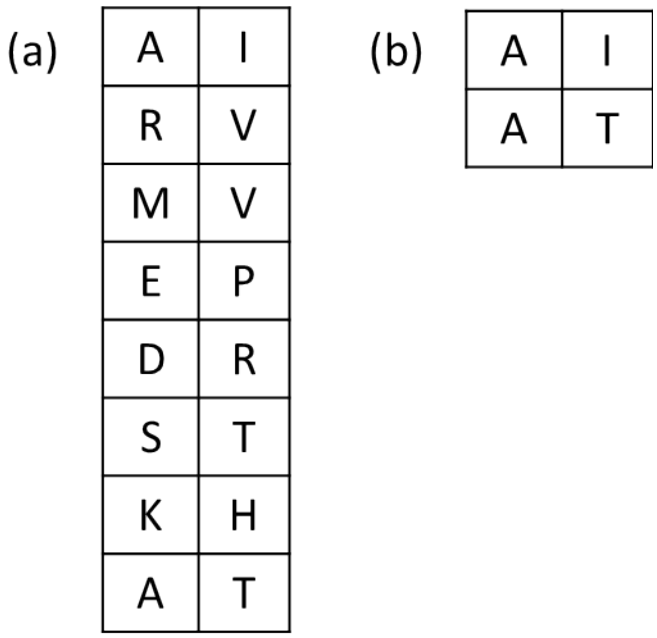

Figure 3.1: How perturbation-based algorithms generally work. (a) The two columns of the MSA we are focusing on. (b) Setting the constraint that the amino acid in the first column must be an 'A', this is the resulting alignment that we would use for calculations on this column pair.

3.1.3 Perturbation-based algorithms

An example of this class of algorithms is the paper by Dekker et al. [38]. Within this paper is an easy-to-follow example which explains how this class of algorithms works in principle. Say we wanted to find the amount of co-evolution between columns i and j. We start by placing a constraint on the amino acids at column i (for example, i must contain a Y) such that we only use those sequences which fulfil that constraint for that column. We can now inspect the characters present at position j, and see if there is now a visible bias toward certain characters at j, given that we are focusing on the subset of the alignment with a Y at i. See Figure 3.1 for an example of how this might work in practice.

One of the earlier examples of this type of algorithm (called Statistical Coupling Analysis, or SCA) was by Lockless and Ranganathan [111]. This method was also used by Suel et al. [175], and by Dekker et al. [38]. Halabi et al. [84] used an updated version of this method.

Dekker et al. [38] came up with a different example of this class of algorithms, called ELSC (Explicit Likelihood of Subset Co-variation), which again looks at the effect of restricting the amino acid distribution at one position on another position's distribution, but uses a different statistic to measure the effect this has.

Advantages

- Dekker et al. [38] said of the approach of Lockless and Ranganathan [111] and Suel et al. [175]: 'the overall strategy of this approach is novel and potentially powerful because its central subsetting operation allows the hypothesis-driven exploration of co-variation occurring in response to freely chosen *in silico* evolutionary "perturbations". For instance, one may choose to subset the MSA according to physicochemical classes of perturbations that are found experimentally to alter ligand binding, allosteric coupling or conformational equilibria' [38, pp. 1565]. This means that we can explore what is driving the process of co-evolution

Disadvantages

- Restricting the amino acid distribution at one position to consider the amino acids now present at a second position, means that we are effectively reducing the number of sequences in the alignment that we are considering. This class of methods, therefore, requires a large alignment to begin with

3.1.4 McBASC

The McBASC (McLachlan-Based Substitution Correlation) category involves looking at the differences in physicochemical properties between the amino acids at the positions we are considering, whether from values calculated ourselves, or from a similarity matrix.

The method by Göbel et al. (called PlotCorr) [75] involves calculating a distance matrix between the amino acids at each of the columns we are looking at, and then comparing these distance matrices between the two positions in a pair. Olmea and Valencia [141] added information to this method on things such as conservation and apolar residues.

The method by Neher [131], unlike the study by Göbel et al. [75], takes into account one individual physicochemical property (e.g. big/small, positively-charged/negatively-charged, etc.) at a time.

The method by Codoñer and Fares [31] also uses similarity matrices.

Advantages

- These methods consider the physicochemical properties of the amino acids in the alignment, unlike the other methods discussed up until this point, which on the whole, only look at the statistical distribution of amino acids at the site pair we are considering

Disadvantages

- With methods such as that of Neher [131], we need to choose the physicochemical property to focus on, but we often do not know in advance which components of molecular co-evolution (e.g. charge) shall be represented in the alignment we are considering. However, when using a similarity matrix as our source of information about physicochemical properties such as in the method by Göbel et al. [75], all of the information regarding how similar the amino acids are is already encoded

3.1.5 PSICOV and DCA

PSICOV was developed by Jones et al. [96]. Independently, DCA was developed by Morcos et al. [124] (see also Marks et al. [115]). We place these two methods within the same category as they both use 'a global statistical model of the MSA' [37, pp. 251].

PSICOV, or Protein Sparse Inverse Covariance [96], involves computing the sparse inverse covariance matrix for the alignment used as input. (The covariance matrix is simply the matrix of covariances between pairs of columns in the alignment.) The rationale behind this is that 'assuming the sample covariance matrix can in fact be inverted, the inverse covariance matrix provides information on the degree of direct coupling between pairs of sites in the given MSA' [96, pp. 185]. The method used to do so is called the graphical lasso method.

DCA, or Direct-Coupling Analysis [124, 115], uses Direct Information, which is a modified version of the Mutual Information equation. However, unlike Mutual Information, this method requires solving equations to find the value of $P_{ij}^{(dir)}(A, B)$ for each pair of amino acids A and B, which are used in the place of the joint frequencies of A and B at columns i and j in the alignment.

Advantages

- DCA weights frequencies of amino acids based on sequence similarity

Disadvantages

- The algorithms within these methods are substantially more intricate than the other methods we have already described, and yet they do not include measures of the physicochemical properties of the amino acids in the input alignment

3.1.6 Sequence divergence-based approximation

Fares and Travers [56] developed a method using sequence divergence-based approximation called CAPS (Coevolution Analysis using Protein Sequences). This method compares the variance of evolutionary rates between two sites in an alignment, and corrects this variance for differences in divergence time between sequences. We have many sequences in an alignment, and these sequences may vary in the time each pair diverged; 'a given alignment can include sequences whose pairwise distance is significantly divergent from the mean pairwise distance' [56, p. 11].

This method was originally classed as parametric in the classification developed by Codoñer and Fares [31] due to there existing an additional subprogram which allows for an extra analysis of the data using information from a phylogenetic tree.

Advantages

- This particular method, when using randomisation removes the impact of $C_{stochastic}$. (This is done by simulating alignments, and testing them according to the CAPS method, which gives us a null distribution which we can then compare against the "real" data)

- Can find which components of co-evolution signal are due to phylogenetic signal by calculating correlations for entire alignment, and then re-calculating the scores after removing parts of the phylogenetic tree (which gives us sub-alignments of the whole alignment). If a pair of co-evolving sites can be detected in the entire alignment but not for a sub-alignment, then those sites are co-evolving simply because they come from related sequences (optional)

- Can use information from the three-dimensional structure of proteins (optional)

Disadvantages

- Affected by amount of divergence between pairs of amino acids in a column
- Sensitivity affected by 'a low number of sequences combined with a high pairwise sequence divergence' [56, pp. 13]
- 'Assumes that the coevolutionary relationship between a pair of sites remains constant through time' [56, pp. 22]

3.2 Parametric methods

Parametric methods differ from non-parametric methods in that unlike non-parametric methods, we require the phylogenetic tree showing the relationships between the sequences in the input alignment. There are many different parametric methods in the literature, including those based on Maximum Likelihood Approximation, tracking changes on branches of the tree, and Bayesian Mutational Mapping. The motivation for including such information in these methods is that it can be argued that in order to distinguish background co-evolution due to the phylogenetic relationships between sequences from "true" co-evolution, we must take account of the tree describing the relationships between the sequences.

3.2.1 Maximum Likelihood Approximation

Maximum Likelihood Approximation (MLA) uses the Likelihood Ratio test statistic to work out whether each pair of sites is more likely to be co-evolving than not. This equation is given as [153, p. 190]:

$$LR = -2ln(L_I/L_D) \tag{3.4}$$

L_I is the maximum likelihood value obtained under the independent model (where the two positions are not co-evolving), and L_D is the maximum likelihood for the dependent model (where the two positions are co-evolving). These are both calculated using information from the phylogenetic tree describing the relationships between the sequences.

The details of the Maximum Likelihood method of identifying co-evolving pairs can be found in the papers by Felsenstein [57], Pollock et al. [153] (which builds upon the earlier work by Felsenstein), and Pagel [143].

As mentioned above, we have an independent model, and a dependent model. The independent model is defined by the parameters:

- λ, which is simply the rate at which changes occur from one character to another
- π_A and π_a, which tell us how likely we are to see characters A and a, respectively, at this column. A and a are actually "states" or categories, so A could mean the amino acid is "large", and a means it is "small" – we need to put the amino acids into categories based on some property

These parameters are defined per column of the alignment, and can be used to work out the probabilities that, for example, character i will change to character j in time t. The model used is a Markov process model following Felsenstein [57], which means that the likelihood of a character changing to any other (or staying the same) does not depend on the "history" of characters at that position.

The dependent model used is a special case of that developed by Pagel [143], which uses the same parameters as above (of course, we need to define these for each of the two columns separately), but the transition matrix is defined differently; we now consider changes between the possible state pairs $\{AB, Ab, aB, ab\}$, where AB, for example, means that the characters at both positions are within the same category (both large amino acids, for example). Along with the parameters in the models just described, a phylogenetic tree is input into this method, and used to calculate the final Likelihood Ratio value.

Advantages

- We do not need to know the phylogenetic tree precisely; 'as long as the tree is approximately correct, the effects on parameter estimation can be minimal' [153, pp. 190] (i.e. the effect on the final results can be minimal)
- We can use any partition of the amino acids into groups

Disadvantages

- The method may be limited to use with 'moderately closely related protein families' [153, p. 188] (although 'moderately' was not defined here), but this should not prove a problem as we are looking for co-evolution in homologous sequences

- We do not know which physicochemical properties may be driving co-evolution for each particular protein (a problem for any method which uses a reduced alphabet). However, we could say that we would use the "full" alphabet instead of a "reduced" alphabet, which would naturally increase the number of calculations we would need to perform

- There are dependencies between which calculations we do and when – we need to perform calculations for the leaves of the tree first and work through the internal nodes. This could be challenging to implement in parallel (although methods of tree traversal have been developed in CUDA)

3.2.2 Tracking changes on branches

The three methods within this category all find co-evolving sites by attempting to work out where the mutations occurred on branches. We describe these in the order in which they were published. None of these methods cite each other, suggesting they were developed independently, but due to their similarity, we have decided to place them in the same category. We describe each of the three methods in turn, and then describe the advantages and disadvantages of this category of methods as a whole.

Fukami-Kobayashi et al. [68]

Fukami-Kobayashi et al. [68] decided to look specifically at node-leaf and node-node comparisons, arguing that because such comparisons on a phylogenetic tree would involve fewer mutations (and hence a shorter total evolutionary distance), we would be more likely to detect stronger co-evolutionary signals that are less obscured by mutations at multiple sites. Like the study by Pollock et al. [153], Fukami-Kobayashi et al. used a reduced alphabet (e.g. placing amino acids into categories of positively-charged and negatively-charged amino acids).

The ancestral sequences of the trees were computed using the Fitch parsimony method [59]. For every branch in every tree, the authors checked to see whether any charge-reversing amino acid substitutions happened with a probability of at least 0.2, that is, for every branch, answering the question: are the amino acids present at the nodes attached to that branch in different "charge groups"? They then found all pairs of trees where, on the corresponding branch on each tree, there was a charge reversal (positive to negative, or vice versa). It was assumed that only pairs of positions that are close in space in the 3D structure of the protein could actually be co-evolving.

This study largely focused on how co-evolution signals can be strengthened through comparing ancestral sequences with other ancestral sequences, or by comparing ancestral sequences with extant ones. This is because the shorter the evolutionary distance between two sequences, the fewer the expected number of mutations between those sequences, and the easier it would be to identify mutations that are occurring as a result of compensatory covariation.

Fleishman et al. [61]

The method of Fleishman et al. [61] quantifies the magnitude of the change from the amino acid at the top of a branch, and that at the bottom. This is done using the differences in polarity and volume between the amino acids found at either end of a branch. Pearson's correlation coefficient is calculated between these distances. After bootstrap sampling is used to repeat the experiments, Principal Components Analysis is used to identify clusters of co-evolving sites.

Waddell et al. [189]

The study by Waddell et al. [189] uses the detection of co-evolution as an "intermediate step" towards solving the problem of finding protein-protein interactions – this is to do with inter-molecular co-evolution, not intra-molecular co-evolution, which we are interested in. However, in the Discussion section of this paper, there is a suggested method which looks at individual positions within a protein. After reconstructing a tree and ancestral sequences for the input alignment, each branch is inspected in turn with the aim of identifying where the mutations are likely to have occurred. Unlike the previously-described methods, this method just looks for changes without paying attention to whether those changes make any difference to the physicochemical properties of the amino acid at that position. These changes are coded in a binary matrix, where a 1 means that a change has occurred at a specific branch, at a specific column of the alignment, and a 0 means such a change did not occur. (If a change occurred it means that, given a branch, at a particular column, the character at that column in the sequences at either end of that branch are different.)

After the binary matrix has been constructed, we may calculate either Mutual Information or Cohen's Kappa [33] for each pair of columns in that matrix. Mutual Information is calculated as we have seen in the Mutual Information equation previously (see Equation 3.1), except our "alphabet" consists of only two states: 0 and 1. Cohen's Kappa is a statistic used to measure the

amount of agreement between two annotators (in this case, between the two columns we are looking at).

Advantages

- As described in the paper by Fukami-Kobayashi et al. [68], smaller distances between sequences should mean more reliable results
- Fukami-Kobayashi et al.'s [68] and Fleishman et al.'s [61] methods allow one to account for the physicochemical properties of the amino acids, whereas we could say that Waddell et al.'s [189] method does not require the definition of a hypothesis as to where the co-evolution originates from (e.g. charge, size)
- Fleishman et al.'s [61] method and Waddell et al.'s [189] method do not require the 3D structure of the protein being examined

Disadvantages

- Each of the above-listed advantages can also be seen as disadvantages
- The Fitch parsimony algorithm used by Fukami-Kobayashi et al. [68] is not intended for use with amino acid sequences, and if we were to use this method, we would need to replace this algorithm with another method adapted from the original, but designed for protein sequences
- The method by Fleishman et al. [61] depends on the mutation matrix being used – the Miyata matrix, which was used in this particular method, only takes into account polarity and volume; other matrices that could be used could take into account frequencies of amino acids in real data (for example, BLOSUM or PAM matrices)
- All of the methods in this category, like other parametric methods, depend on the quality of the tree reconstruction method being used, and also the ancestral sequence reconstruction method being used

3.2.3 Bayesian Mutational Mapping

The BMM (Bayesian Mutational Mapping) method by Dimmic et al. [40] uses an approach based on phylogenetic trees and Bayes' theorem to detect co-evolution. It uses the following hypothesis: if two sites are co-evolving, then a mutation at one site will influence the rate of substitution at the site that it is co-evolving with. BMM allows us to choose which test statistic we use to test our hypothesis (both parametric and non-parametric test statistics are

supported). It should be noted that in contrast to the other methods described in this chapter, this method requires the use of nucleotide (DNA) sequences.

It should be noted that we were going to consider this as part of the comparison of the performance of various co-evolution detection methods in Chapter 6, however we were not able to run the original program to allow for verification of our results by comparison to the results from the original program, so this method was abandoned.

Advantages

- By sampling over the respective distributions of the different parameters we are using, we do not need to be concerned that, for example, we have chosen the "correct" tree to use
- Have a selection of different test statistics to choose from

Disadvantages

- We make the assumption that nucleotides in a codon are independent of each other (as mentioned above, this algorithm uses nucleotide sequences). This means that when we do the site simulation, we are not using a true null model
- This method would probably be the most complex to implement in parallel

3.3 Discussion of methods for detecting molecular co-evolution

So far in this chapter we have given an overview of the main classes of method for detecting molecular co-evolution. Here we first discuss the use of different alphabets, and the parallelisation potential of the methods generally. Then we compare the methods and decide which to implement in the next chapter, in a biological performance comparison.

3.3.1 Alphabet choice

Of the methods discussed, most use a full alphabet, and a couple use a binary alphabet, or some other alphabet where we put amino acids into categories based on physicochemical properties.

The study by Caporaso and colleagues [22] investigated the use of different kinds of alphabets when identifying co-evolutionary signals. They found that using the correct reduced alphabet ("reduced" being what we describe as categories of physicochemical properties – the 'correct' one would be the one which explains correctly why co-evolution is occurring, e.g. if it were to preserve charge, we would choose an alphabet segregating amino acids based on their charge) increases the power of analyses, but that using an alphabet which does not represent well 'the primary biochemical property subjected to coevolutionary pressures by natural selection' [22, para. 5] decreases power. Considering we shall be using our algorithms for many different and diverse proteins, and that we do not know the main driver for co-evolution for each of these proteins ahead of time, it is wise to use a full alphabet for our analyses.

3.3.2 Parallelisation potential

To the best of our knowledge, only one parallel implementation of a method for detecting co-evolving sites has been published. This is the Mutual Information method of Waechter et al. [191]. Generally speaking, non-parametric methods would be easier to implement for the CUDA architecture than parametric methods. This is due to the fact that array and matrix-like data structures are easier to process on the CUDA architecture than tree-like data structures, as the main operations of a CUDA graphics card relate to n-dimensional arrays.

In the next section, we compare the methods to each other mostly from the perspective of biological performance, although we will also consider the parallelisation potential of each of the methods.

3.3.3 Method comparison

Before we begin a comparison of the different categories of method described in the previous sections, we note that non-parametric methods have been used overwhelmingly more than parametric methods in previous studies. This means that there are a lot of papers reviewing and comparing the same few methods such as MI and McBASC, and fewer papers looking at parametric methods. Each method's name shall be the one given in the paper being discussed, but there are some differences between authors, which we shall note on a case-by-case basis. Also, sometimes the definitions of such terms as sensitivity, specificity, and accuracy, do not seem to mean what we expect them to mean, so we also give the definitions used by the authors.

Most of the comparison papers tend to give results in terms of the number of correct predictions of co-evolving sites compared to a known 3D protein structure. For ease of explanation, we refer to these studies as "predicted pairs to structure comparisons". Also, some of the papers analyse the results by looking at each protein family individually, and ranking the correlation scores from the highest-scoring pair to the lowest-scoring pair. They then consider only the top 1 pair, then the top 2 scoring pairs, and so on, examining what effect this has on accuracy, etc. We call these studies "rank-order studies".

We now summarise the results of several review papers.

Fodor and Aldrich [65]

- compared MI [10], OMES [99], SCA [111], and McBASC [140]
- on a real data set of 224 protein families
- "predicted pairs to structure comparison" and "rank-order study"
- overall, McBASC and OMES had similar accuracy[1], McBASC does slightly better as the number of highest-scoring pairs that are considered is increased

Dunn et al. [42]

- compared MI, MIr [116], MIp, MIa, OMES [65], and McBASC [65]
- on 83 protein families
- "predicted pairs to structure comparison" and "rank-order study"
- the best precision[2] was obtained with MIp and MIa
- what is interesting here is the difference that normalising MI has on results; without normalisation (as in the study by Fodor and Aldrich [65]) McBASC appears to be the best method

[1] 'Accuracy is defined by CASP as the number of residue pairs correctly predicted within 8 [Angstroms] divided by the number of residue pairs submitted for evaluation.' [65, pp. 218] (CASP is a competition for protein structure prediction [187].)

[2] The statistic was described as 'the fraction of pairs at each rank or higher that were in contact in a representative structure' [42, pp. 337], which we call precision.

Ackerman et al. [1]

- compared MI, ZPX [74], dbZPX2, dgbZPX2, nbZPX2 (these three having been developed by the authors themselves) (ZPX, dbZPX2, dgbZPX2, and nbZPX2 are all different ways of normalising MI), logR [18] (called B&vN by Jones et al. [96]), DCA [124, 115], OMES [65], McBASC [65], ELSC [38], SCA (fodorSCA [38], and ramaSCA [84]), and PSICOV [96]
- on simulated data generated by (their own) MSAvolve (Multiple Sequence Alignment (e)volve) simulation program, and then by the SIMPROT (Simulation of Protein sequences) program
- it was found in both cases that the best-performing methods were the normalised MI methods (ZPX2, dbZPX2, dgbZPX2, nbZPX2), DCA and logR
- a similar pattern was found in two data sets of real data (this was a "predicted pairs to structure comparison"), although PSICOV could only be tested for the larger data set of real data – in this case, it performed better than logR
- normalising MI makes a difference to the relative performance of the methods

Jones et al. [96]

- compared their own method PSICOV, the same method without sequence weighting (sequence weighting meaning related sequences are down-weighted relative to more unique sequences in the alignment) (PSICOV - SW), MIp [42], MIp with sequence weighting (MIp + SW), and the Bayesian method of Burger and van Nimwegen [18] (B&vN) (called logR elsewhere)
- on 150 protein families
- "predicted pairs to structure comparison"
- the ordering of the different methods in terms of their average precision is: PSICOV, PSICOV - SW, B&vN, MIp + SW, MIp
- the methods approach similar performance as we consider more and more top-scoring pairs ("rank-order study")

Fares and Travers [56]

- compared their own method, CAPS, MI [103] (this method was referred to as MICK within the comparison), Dependency (a way of normalising MI) [181], and lnLCorr (called Maximum Likelihood Approximation elsewhere) [153]
- on simulated data
- the formulae that Favers and Travers gave for sensitivity and specificity actually represent positive predictive value (precision), and negative predictive value, respectively
- for 'sensitivity', CAPS performed a lot better than the other methods
- for 'the percentage of true positive covariation pairs' [56, pp. 14], the performances of different methods can overlap for a given number of sequences per alignment
- CAPS seems to have the most consistent performance overall
- CAPS also has the highest 'specificity' of the four methods

Fuchs et al. [67]

- compared McBASC (with two different substitution matrices, Miyata and McLachlan, called McBASC-Miyata and McBASC-McLachlan) [141], two versions of OMES (OMES-KASS [99], and OMES-FODOR [65]), CORRMUT (which uses the same method as Fleishman et al. [61]) [60], CAPS [56], MI [73], SCA [111], and ELSC [38]
- "predicted pair to structure comparison" and "rank-order study"
- on 14 membrane proteins (note that CORRMUT and CAPS could not generate results for all 14 proteins), accuracy[3] is (in order of highest to lowest, with the accuracy itself given in brackets): McBASC-McLachlan (9), McBASC-Miyata, OMES-KASS (8), OMES-FODOR, CORRMUT, CAPS, ELSC (7), MI (5), and SCA (3)
- on alignments of 'experimentally determined transmembrane segments' [67, pp. 3317], accuracies became: CAPS (12), McBASC-McLachlan, ELSC (10), McBASC-Miyata, OMES-KASS (9), OMES-FODOR, MI (8), CORRMUT (7), and SCA (3)
- CAPS, the McBASC methods, and the OMES methods seem to be the best-performing, depending on which data source is used

[3] Defined as 'fraction of correctly predicted contacts out of all correlations found' [67, pp. 3315]

Gulyás-Kovács [83]

- to the best of our knowledge, this is the only paper which looks at CoMap
- assumes that we already have some knowledge of pairs of positions that are in contact in the 3D structure of a protein
- compared the methods CoMap and many variations on this [44, 43] (these are related to [202, 168, 40]); MI [103], MIp [42], CAPS, and CAPS-t [56]
- on an alignment of 553 sequences
- CoMap was the best method in terms of receiver operating characteristic (ROC) curves, although the methods do not appear to have too much difference in performance
- then including combinations of different optimisations such as alignment filtering, etc.
- for the different optimizations, CoMap is always the best, followed by MIp, (and then CAPS and MI, which can overlap/have different orders)
- for a lower specificity value, MIp does better than CoMap for no optimizations

3.3.4 Methods summary

We now summarise each method previously described in this chapter, along with its advantages and disadvantages.

Mutual Information (MI)
Basic equation from Shannon's Information Theory (i.e. is basically a Computer Science method), takes into account frequencies of characters at each position, and joint frequencies across the pair of positions
Easy to understand, attempts made to include information on physicochemical properties, can be normalised by components of the equation such as joint entropy to improve performance
Makes no distinction between amino acids (treated as characters with no meaning), taking into account physicochemical properties means deciding which "component" of co-evolution is at work, which may differ betweeen proteins and we probably do not know this for sure

Observed Minus Expected Squared (OMES) Equal to χ^2 statistical test, where we use information on the expected and observed frequencies to calculate a statistic Easy to understand Does not use information on physicochemical properties
Perturbation-based methods Consider only certain sequences in the alignment such that the amino acid present in the first column is the same throughout. Calculate a statistic based on what we see in the second column, given that the first column is now effectively conserved This method "makes sense" considering what we know about correlated sites, and was designed for detecting co-evolution, unlike some other non-parametric methods Does not use information on physicochemical properties, only considering certain sequences based on the amino acids in one column can severely restrict the number of sequences we are considering – a large initial number of sequences is necessary
McLachlan-Based Substitution Correlation (McBASC) Using the physicochemical distances between the amino acids within each of the columns being considered, compute a distance matrix for each column. A statistic is calculated using such distance matrices for a pair of columns Uses physicochemical properties of the amino acids, can choose which substitution matrix to use as source of physicochemical information on amino acids Certain methods within this category depend on the user choosing which physicochemical property they believe is "driving" the co-evolution between pairs of positions, which would likely not be known in advance
PSICOV (Protein Sparse Inverse Covariance) and DCA (Direct Coupling Analysis) These methods were placed within the same category due to the heavier amount of statistical theory behind these methods. PSICOV relies on the concept of a sparse inverse covariance matrix, whereas DCA uses Direct Information, which is like Mutual Information, but finding the joint frequencies of amino acids requires more calculations (they are not just directly taken from the input alignment) Behind these methods there is a lot of statistical theory that conceptually "makes sense"

More complex, yet neither of these methods include information on the physicochemical properties of the amino acids
Sequence Divergence-based Approximation Compares the variance of evolutionary rates between two sites in an MSA, correcting this variance for differences in divergence time between sequences Can remove co-evolution signal due to randomness, can separate out co-evolution due to sequences in MSA being related, optionally can use information on three-dimensional structure Affected by amount of divergence between pairs of amino acids in a column, affected by 'a low number of sequences combined with a high pairwise sequence divergence' [56, pp. 13], 'assumes that the coevolutionary relationship between a pair of sites remains constant through time' [56, pp. 22]
Maximum Likelihood Approximation (MLA) Compares the likelihood of the real sequences evolving under a co-evolutionary model, and under an independent model (with no co-evolution between sites) Makes sense from a biological point of view, do not need to know the exact tree Have to decide how to divide the amino acids into physicochemical-related categories (probably will not know what property may be driving co-evolution)
Tracking changes on branches This category works as expected – it involves attempting to mark on the phylogeny where the mutations probably occurred, and from this try to look for co-evolving pairs More reliable results (at least in theory) as we are looking at changes along each branch independently of the others, rather than trying to look for changes between two leaves in a tree, can decide which information you want to include (different methods within this category can allow for protein 3D structural information, physicochemical properties) Methods depend on the quality of the tree reconstructed (and especially on the quality of the ancestral sequence reconstruction algorithm)
Bayesian Mutational Mapping (BMM) Testing the hypothesis that if two sites are co-evolving, a mutation at one site will influence the rate of substitution of the other. Can use different test statistics, iterate over lots of values of the different parameters, so do not need to worry if these are correct

Assumes nucleotides in a column are independent – this is not true even if the sites are not co-evolving

Table 3.1: The different methods for co-evolution detection.

Table 3.1 shows a comparison of the different methods we have described according to different criteria.

3.3.5 Recommendations

According to a study by Clark and colleagues [30], there may not be one "best" method for all proteins (from a biological perspective), due to variations in structure and evolutionary history. A study by Caporaso and colleagues [22] suggests that tree-aware (parametric) methods are not necessarily better than tree-ignorant (non-parametric) methods, although they did only test and compare these methods on two specific proteins. It is logical to think that tree-aware methods would be better, due to their use of phylogenetic trees to remove spurious signals due to shared ancestry.

Our choices regarding which non-parametric methods to use in our own experiments were made as follows:

- The Mutual Information methods are readily parallelisable, as these methods have been implemented in CUDA already. Using Z-scores to evaluate the signal against the background would also be easy as this just involves running the same calculations over and over. Including biological information is relatively easy; the amino acid background distribution and possible categorisations are freely available. We have decided to use this method as-is, however, as a kind of benchmark method, due to the fact that it is described and used so widely. We chose this method instead of OMES as although they are both methods which only account for frequencies of characters within columns of the alignment, MI is much more widely used.

- We chose not to use one of the perturbation-based methods because unless one has a large number of sequences, restricting the amino acids at one position could give us only a very small number of sequences to work with, if the column we are focusing on is highly variable, for example. Also, looking further into the future, we are not sure if this kind of method would be easily parallelised.

- We chose to use McBASC as although it is a relatively basic method, it does include information on amino acid physicochemical properties

from a substitution matrix, such as PAM or BLOSUM. Its simplicity in combination with this means it can be viewed as "one step up" from Mutual Information.

- Although PSICOV and DCA seem very sophisticated in terms of the statistical models they use, this sophistication means complexity in the implementation, something which we believe would not be conducive to parallelisation using CUDA.

- Because the algorithm in its associated paper and the manual are easy to read, we think that the Sequence Divergence-based Approximation algorithm would be relatively easy to implement; also it fares well in comparison to other methods.

Now moving onto the parametric methods, here is how our choices were made regarding which to include in our experiments:

- Maximum Likelihood Approximation (MLA) is not desirable due to the fact that it uses a reduced alphabet, which would not be appropriate for us, as we have discussed.

- The approach used in Bayesian Mutational Mapping is flexible in that it allows us to use any test statistic we want. We did attempt an implementation of this method. Unfortunately, we could not compare the results of our implementation against the original implementation as the original program would crash with any input other than the example files provided by the author.

- The "Tracking changes on branches" category of methods is intuitive in how it works; we chose to implement the previously-unimplemented method by Waddell et al. [189] due to the fact that, unlike the methods by Fukami-Kobayashi et al. [68] and Fleishman et al. [61], it does not require the user to decide on the likely origin of the co-evolution signal (such as amino acid size or charge) when running the method. We wanted to choose at least one parametric method; the other parametric methods were not appropriate, as discussed.

As a general point, when implementing an algorithm which requires an ancestral sequence reconstruction, there are two main possibilities: Maximum Likelihood Approximation (MLA) and parsimony-based methods. The study by Fukami-Kobayashi and colleagues [68] says that Fitch's parsimony method is not as good biologically, but quicker. MLA is the opposite (better biologically but slow in comparison).

In the end we decided to pursue:

- the original Mutual Information equation (i.e. with no additions) (MI) [191]
- the Sequence Divergence-based Approximation method (CAPS) [56]
- the McBASC method (PlotCorr) [75, 141]
- one of the "Tracking changes on branches" methods as two different methods, henceforth called Waddell – MI and Waddell – Kappa, for the two different test statistics we used with this method, as suggested in the original paper [189]

To ensure a fair comparison, we decided to choose a type of background noise removal to be used across all of the co-evolution detection methods; now we shall overview the main categories of background noise removal, and choose one.

3.4 Methods for removing background noise

Now that we have chosen the basic methods for detecting molecular co-evolution that we shall be comparing, we must choose how we shall remove the background noise from our co-evolution signal between pairs of columns. As discussed previously, co-evolution has several components, some of which appear due to factors such as having a limited number of sequences in the alignment, the fact that the sequences are related to each other, and just due to chance. Each of the co-evolution detection methods we have discussed in this chapter had their own method for removing background noise, but we decided to choose one method for noise removal that we would use across all of the co-evolution detection methods. In this section we compare several of the most common methods for noise removal, before deciding on one category of method to use throughout.

Dutheil [45] gives several options for what we can do to remove the influence of the fact that the sequences are related (i.e. removing $C_{phylogeny}$): doing nothing at all; alignment pre-processing (curation) before running our methods; correcting for the influence of phylogeny in the null hypothesis of no co-evolution; and correcting for the influence of phylogeny in the test statistic itself. We ignore the first option (doing nothing). We shall now discuss the other three options in more detail.

3.4.1 Alignment curation

This basically involves processing the alignments somehow in advance of running the co-evolution detection methods, with the aim to remove as much of the background phylogenetic signal as possible before even running the methods.

3.4.2 Correcting the null hypothesis/test statistic to account for phylogeny

The approaches "correcting the null hypothesis to account for phylogeny" and "correcting the test statistic to account for phylogeny" are somewhat linked; in particular, many of the examples given in the paper by Dutheil [45] have methods whose approach to removing background noise corrects both the null hypothesis and the test statistic, so we present these together. These approaches can be further split into three categories of approach: non-parametric bootstrapping, parametric boostrapping, and Z-scores. We shall give a couple of examples of methods from each of these categories to illustrate how they work generally.

Non-parametric bootstrapping

Bootstrapping involves using the input alignment to build, or simulate, new alignments. Non-parametric bootstrapping means that we do not use any parameters to build the new alignments. This can mean we do something as simple as "shuffling" the alignment. For example, Caporaso et al. [22] shuffled the columns around (changed the ordering of the columns in the alignment) to build "new alignments". In this case, these shuffled alignments were used as examples of alignments where there should be no co-evolution due to structure ($C_{structure}$). Another example of a non-parametric boostrapping approach is that used by Waechter et al. [191], where they again shuffled their alignments, but this time they did the shuffling on a per-column basis, meaning they mixed up the characters within a column, while keeping the columns themselves in their respective positions.

Parametric bootstrapping

This uses bootstrapping, but also includes parameters too. For example, Wollenberg and Atchley [199] used a phylogenetic tree constructed from the real data in combination with a substitution matrix to generate new data sets. The correlation values (MI in this case) were used to generate separate distributions of the correlation values for the real data, and for the simulated data.

The idea is that the simulated data should only have noise derived from the sequences being related, and chance. The simulated data distribution was then used to derive a threshold MI value above which, for the real data, the authors would determine that the correlation may be biologically significant.

Z-scores

Z-scores give, for a given value, how far away that value is from the mean, in units of standard deviations. The equation is:

$$Z = \frac{x' - \overline{x}}{\sigma(x)} \tag{3.5}$$

We have a set of values, x. The mean of the values is given by $\overline{x}$, and the standard deviation is $\sigma(x)$. An individual value in this set is given by x'.

3.4.3 Chosen method of background removal

We have decided to use Z-scores to remove the background noise due to the fact that Z-scores can be used in conjunction with non-parametric bootstrapping, as we shall discuss below in more detail. We decided against parametric bootstrapping as we wished to keep non-parametric methods non-parametric; parametric bootstrapping requires the reconstruction of a phylogenetic tree for the input data, which defeats the purpose of using non-parametric methods in the first place. We now discuss our Z-score methods in a lot more detail.

The use of Z-scores for background noise removal is what unifies all five of the co-evolution detection methods to be compared. There are two approaches to calculating the Z-score. We call these the shuffle method and the original values method. The shuffle method uses information from the columns we are looking at to remove background noise, whereas the original values method uses information from the entire alignment whenever we want to calculate the correlation of any pair of positions.

The original values Z-score approach can be found in several papers (for example, [73] and [42]). The shuffle method was found in the paper by Waechter et al. [191]. We aim to see what difference each of these Z-score approaches would make to the results. We believe that the original values Z-score method would be better than the shuffle Z-score method because, rather than just removing background correlation within a column (which the shuffle Z-score aims to do), we are removing the background from the rest of the alignment. That said, these two methods have not, to our knowledge, been compared ex-

perimentally, which we shall do in Chapter 6. We now discuss each of these Z-score approaches in more detail.

Shuffle

We shall describe how this approach would work for Mutual Information (MI), but note that the general method is also the same for CAPS and PlotCorr. We then describe the amendments that need to be made so that this approach can work for the parametric methods Waddell – MI and Waddell – Kappa.

This method involves shuffling all of the characters in each column of the input alignment in place, and then re-calculating the MI for each pair of columns (this approach uses non-parametric bootstrapping). Repeating these two steps some number of times (we used 100), we can use the mean and standard deviation for the shuffled MI scores to calculate the Z-score for the actual MI value:

$$Z_{i,j} = \frac{x_{i,j} - \overline{\text{shuffled } x_{i,j}}}{\sigma(\text{shuffled } x_{i,j})} \tag{3.6}$$

$\overline{\text{shuffled } x_{i,j}}$ and $\sigma(\text{shuffled } x_{i,j})$ are the mean and standard deviation, respectively, of the MI (or other correlation statistic) values calculated for column pair i and j for the shuffled versions of the alignment. $x_{i,j}$ is the correlation value for that pair i and j before we did any shuffling at all (i.e. for the original alignment). An example using the shuffle Z-score approach is given in Appendix A.1.

The shuffling of each column in place removes background noise from the overall correlation between positions by breaking the associations between amino acids within a sequence.

As mentioned above, the only difference between the non-parametric and the parametric methods for this Z-score approach is in how the matrices are shuffled. In the parametric methods, instead of calculating statistics based on our alignment directly we are working with binary matrices. Shuffling these in the way described above would eliminate the meaning of the matrices themselves. This is because each cell in the matrix tells us whether there is a difference between the sequences at either end of a branch of the phylogenetic tree, this tree representing the relationships between the sequences in the input alignment. Shuffling the columns of the binary matrix would mean that the totals for each row of the matrix would change.

Because of this, we have a different method of shuffling the matrix. The algorithm for permuting the matrix was adapted from code given in FMc's answer on a StackOverflow question [172]; we adapted the code from Perl to C++ (code on StackOverflow is available under the Creative Commons CC BY-SA license). This binary matrix shuffling code works by first picking one value in the entire matrix to be inverted (1 to 0 or 0 to 1). This upsets the row and column totals, so it then fixes the totals by making further changes to the matrix. This allows us to keep the row and column totals of the binary matrix constant.

To determine whether one call to this randomisation function would be enough, we looked at the correlation between the original matrix and each of the generated matrices, for several examples. For each example alignment, we generated 1000 shuffled alignments for the non-parametric methods, and 1000 of the corresponding binary matrices for the parametric methods, and measured the amount of correlation between each new matrix and the original matrix. From this we found that shuffling the matrix once was enough.

Original values

There is an alternative way to calculate the Z-scores, and this was found in a couple of papers [73, 42]. This approach involves using the original values of whichever test statistic we have calculated for every pair of columns. We calculate the mean and standard deviation of these values, and can then straight away calculate the Z-scores for each pair of columns. An example using the original values Z-score approach is given in Appendix A.1.

This type of Z-score removes background noise from the overall correlation by finding a mean correlation for all possible pairs (which we take to be the background noise), and removing this (subtracting it) from an individual pair's correlation value.

We have decided to compare the original values and shuffle Z-score types in the next chapter; although we have said we believe the original values Z-score would perform better than the shuffle Z-score, this has not, to our knowledge, been shown experimentally yet.

3.5 Summary

In this chapter we have compared different categories of co-evolution detection method, and also different approaches toward removing background

noise. We have chosen the methods to implement, which we shall compare on simulated data. We wish to ensure that our simulated data is similar to the real data we shall be using our methods on in the future, so in the next chapter we discuss data collection.

Chapter 4

Data Collection

We wish to compare the methods we selected in the previous chapter according to how well they can detect co-evolving sites within alignments. We could use real data to do this, but we do not know for certain where the true co-evolving sites are; indeed, sites which are correlated (i.e. which have character pairs which occur more often in a pair of columns than expected by chance) are only hypothesized to have occurred due to molecular co-evolution [176]. Because of this, we wish to simulate our test data, simulating the effects of co-evolution, so that we know for certain where the truly co-evolving sites are and can evaluate our algorithms' performance in identifying these sites (without false negatives or false positives). Before we can do that, though, we must ensure that the data we are simulating is similar to the real data we shall apply our methods to; in this chapter we shall detail how we gathered that real data. We want multiple sequence alignments of human sequences with sequences from other species. In the next chapter, we will derive simulation parameters from this data. This ensures that our simulated data emulates the important characteristics defining real data. In Chapter 8, we will analyse the real data to identify co-evolving sites.

This chapter starts by describing the choice of database for the data collection, and then the methodology used to obtain the data itself. The steps this chapter covers are shown in Figure 4.1.

4.1 Databases

Before actually collecting the data, we need to choose which database to use. In this section, we first review some of the possible databases we could use, and shall then detail the reasons for our choice. For each of these databases, we detail how it approaches constructing the relationships between proteins.

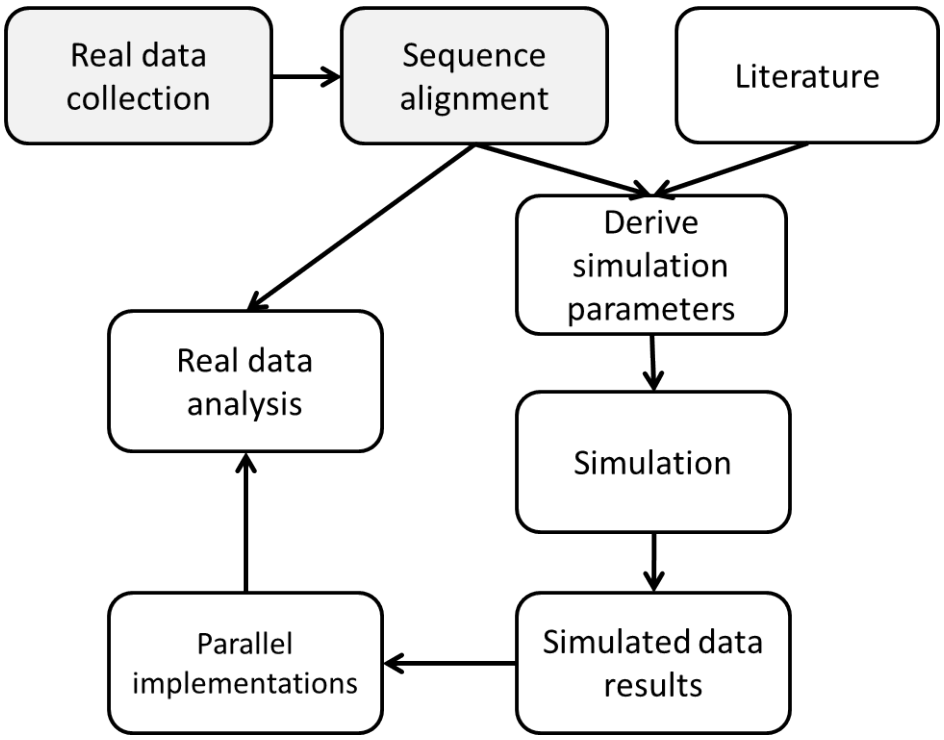

Figure 4.1: This chapter covers the real data collection from the database HomoloGene, and alignment of these sequences.

4.1.1 Ensembl

The Ensembl database stores genome sequences from 86 vertebrate species [52] from a variety of sources [63]. It is managed by the European Bioinformatics Institute and the Wellcome Trust Sanger Institute [51]. The method Ensembl uses to find the ancestral (homology) relationships is given below, summarised from their webpage on the topic [54]:

1. Take all gene sequences stored in the Ensembl database. Find a translation of each gene sequence [1].

2. Use BLAST to find which sequences are related to each other based on how they look. BLAST (Basic Local Alignment Search Tool) is a tool used to search a database for sequences similar to a provided query sequence.

3. Generate clusters of gene sequences based on similarity relationships found in the previous step.

4. Split particularly large clusters (ones with >400 sequences) into smaller ones.

[1] A translation is where we take the DNA sequence, and then turn it into the appropriate protein sequence. This is as opposed to beginning with protein sequences obtained through experimental means.

5. Build a multiple sequence alignment of each cluster.

6. Build a phylogenetic tree for each alignment.

7. Use each phylogenetic tree to find orthology and paralogy relationships (between and within species, respectively) between pairs of genes.

8. Give each GeneTree an ID.

4.1.2 HomoloGene

HomoloGene is a database of homology relationships between protein sequences managed by the NCBI (National Center for Biotechnology Information) [130]. There are 21 species, of a mixture of vertebrates and invertebrates, represented in this database. The method for constructing the homology relationships, summarised from NCBI's webpage on the build procedure [129], is:

1. Compare protein sequences using BLAST.

2. Build a tree from sequence similarity.

3. Use the tree to put sequences into groups, beginning at the leaves of the tree and working towards the root.

4. Map protein alignments back to their corresponding DNA sequences.

5. Calculate distance metrics for the sequences generated from the previous step.

6. Match sequences using synteny where applicable [2].

7. Match up any remaining sequences.

8. Find orthologous and paralogous relationships using set criteria.

4.1.3 UCSC Genome Bioinformatics

UCSC Genome Bioinformatics is managed by the University of California, Santa Cruz. When we visited this website in late 2014, we found the set of multiple sequence alignments of seven vertebrate species (including human) for the protein coding sequences of their genomes (since then alignments with more species have been added).

[2] Synteny is 'a term used to describe the state of two or more genes being present on the same chromosome, though not necessarily [inherited together].' [128]

The multiple sequence alignments were constructed as follows [185]:

1. Use LASTZ to generate pairwise alignments between each of the six non-human species and human

    - LASTZ is the newer form of BLASTZ; the only difference is that bugs encountered in BLASTZ have been fixed in LASTZ [23]
    - The version of BLASTZ used by UCSC derives ultimately from Gapped BLAST, which extends the original BLAST to allow the construction of alignments containing gaps [6]. A separate implementation of this program formed the original BLASTZ [166] – the main difference between this and Gapped BLAST is that it 'essentially guarantees that computer memory will never be a constraining resource' [166, pp. 585]. Some further improvements to the algorithm used gave the version of BLASTZ that UCSC uses [165].

2. Link pairwise alignments into chains; a chain is 'an ordered sequence of traditional pairwise nucleotide alignments ("blocks") separated by larger gaps, some of which may be simultaneous gaps in both species' [100, pp. 11485]

3. Each chain is given a score in a similar way to how alignments are given scores, and the chains are ordered based on their scores, from maximum to minimum. Chains are taken from this ordered list in turn and arranged on the genome, giving us a "net"

4. Pairwise alignments produced (we assume in the following step) are aligned together to produce multiple sequence alignments

4.1.4 Choice

First we shall analyse some of the similarities and differences between the databases just described (Ensembl, HomoloGene, UCSC), and then detail why we chose to use the database we did. The first major difference between each database is the number of species represented in each; Ensembl has 86 vertebrate species, HomoloGene has 21 of a mixture of vertebrates and invertebrates, and UCSC has 7 vertebrate species. 7 is quite a small number in comparison with the others, so we do not consider UCSC as a possibility (after time of writing, it should be noted that the number of species in alignments produced did increase).

Comparing the methodologies used to construct the datasets within Ensembl and HomoloGene, it is apparent that these methods are quite similar. One difference is that Ensembl translates gene sequences into protein sequences, while HomoloGene begins with protein sequences and later back-translates to gene sequences. Both methods begin by using BLAST to find relationships between sequences, and then construct either clusters (Ensembl) or a tree (HomoloGene) based on these relationships. Ensembl does not use any further distance metrics (it just finds homology relationships using a tree constructed from the clusters), whereas HomoloGene uses the concept of synteny and distance metrics to find the relationships. Both databases produce orthology and paralogy relationships between the sequences as the final output.

We have decided to use HomoloGene as our database of choice – it has a balance of enough species and also uses more criteria for finding homology relationships than Ensembl does. This does not necessarily mean that the relationships are "more reliable", but for the sake of making an initial choice on which database to use, it is sufficient.

4.2 Data collection methodology

We shall now outline how we collected the "real" data from HomoloGene, and then how we made our alignments.

HomoloGene provides us with clusters, each one containing a set of homologous sequences from evolutionarily related species. We wanted to generate alignments for the clusters which contained at least one human sequence. This is the method we used to do so:

1. For each HomoloGene cluster ID (only including clusters which have at least one human protein/gene):

   (a) Retrieve the HomoloGene-made protein sequence alignment for that cluster.

   (The next steps are necessary due to the fact that HomoloGene is updated less frequently than the other databases run by NCBI, and the sequences used to construct the HomoloGene alignment may have been updated, or deleted (from NCBI's Protein database), since the alignment retrieved in the previous step was constructed.)

   (b) For each protein ID in that alignment:

i. Retrieve the XML file for that protein from NCBI's Protein database.

ii. Check the status of the protein in the XML file we just retrieved. There are three possible statuses: live, replaced, and suppressed.

- Live: we do not need to do anything further; if the ID is live then that means it is still in use. Use the sequence from the HomoloGene alignment for this protein.
- Replaced: look up (in the same XML file) which ID this one has been replaced with, and retrieve the protein sequence for the new ID.
- Suppressed: this means it has been removed, so we need to go back and look at the gene ID associated with that protein; we can retrieve this by obtaining a slightly different type of XML file, and looking for the associated gene ID. Once we have the gene ID, we need to check if that gene ID is live. If so, continue. If not, we either find the gene ID it has been replaced with or skip this sequence altogether if we cannot find a replacement.
- Assuming we have some gene ID to look at for a new protein sequence, we then check whether our original protein ID had any identical proteins associated with it. If so, check these against the protein IDs associated with our gene ID; if there is a match, use that sequence from that protein ID. If there is not a match or there are not any identical proteins to begin with, then pick the first protein ID available to us, and retrieve the protein sequence from that. If there are no protein IDs for our gene ID, skip this.

(c) Use T-Coffee's M-Coffee feature (described below) to align the sequences [136], and then filter out columns with a threshold of 4, as this is the default value used by the M-Coffee server [26].

4.2.1 Alignment

We were originally going to align our sequences using some alignment program, and then (with the recommendation of Richard Emes, Professor of Bioinformatics at Nottingham University (personal communication, 23rd October 2015)), we were going to use Jalview [194] to check each position in the alignment generated for its quality – aligning amino acids so that they appear in the same column of a multiple sequence alignment implies that these char-

acters descended from an ancestral character [112], but when the sequences we are aligning are of different lengths, we may have several different possibilities for how to align those sequences [108]. When we inspected the Jalview program, we noted that it offered several different multiple sequence alignment programs, all in the class of progressive aligners, which means they cluster together pairs of sequences (as well as the resulting pairwise alignments with other sequences) according to relatedness, and align progressively, rather than aligning the entire group of sequences in one go [112]. All of these were described in the review by Löytynoja [112, pp. 215] as follows:

- T-Coffee: 'Original consistency-based progressive aligner; meta-aligner' [112, pp. 215] – consistency-based aligners use an objective function which attempts to find the alignment between two sequences, A and B, which has the maximum consistency value when we look at 'A and B's independent alignment to outgroup sequences C, D, E, etc.' [112, pp. 218]

- ProbCons: 'Probabilistic variant of the consistency algorithm; amino-acid sequences only' [112, pp. 215] – such probabilistic variants compare the alignment of A and B against all possible alignments, weighting them using their posterior probabilities

- MUSCLE: 'Fast progressive aligner with iteration and refinement' [112, pp. 215] – 'iteration and refinement' means we go through an iterative process whereby we split our initial alignment produced into two smaller alignments and then re-align them together; 'if this alignment is better than the previous one, it is accepted and the process continues' [112, pp. 218]

- MAFFT: 'Fast progressive aligner with iteration and refinement using consistency score' [112, pp. 215]

- ClustalW: 'Classical progressive aligner' [112, pp. 215]

From this list of alignment programs we decided to investigate T-Coffee, its sister program M-Coffee, as well as Clustal Omega as we had come across this program frequently before. The quality score used by T-Coffee and M-Coffee (and Coffee's other "flavours") is called TCS. One of the weaknesses of other scoring methods is 'their reliance on sequence identity and their tendency to exclude phylogenetically informative sites containing too many indels' [27].

Clustal Omega [169] uses a progressive alignment approach, and uses a relatively fast method mBed to produce the guide tree for the alignment. The

alignments are then made using HHalign. Clustal Omega performed well at being able to find alignments when compared to T-Coffee in benchmark tests, but worked out to be a lot slower (e.g. the second benchmark took under half an hour to complete in Clustal Omega, and about 2 days in T-Coffee) [169].

It should be noted that the paper introducing and outlining T-Coffee [136] was published 11 years before that for Clustal Omega [169]; as such, the comparisons made in the former paper are to older versions of Clustal.

M-Coffee (where the "M" stands for "meta") takes T-Coffee's original concept of combining lots of alignments from a library and extends it to combining 15 alignments from different alignment programs. In the paper introducing this method, it was found to be an improvement on the individual methods whose alignments provided information to M-Coffee [192]. However, it is not necessary to include the results from all 15 considered alignment programs in order for M-Coffee to work well. In fact, it is recommended that this is not done, as it only works well as long as relatively dissimilar methods are included in the consensus. It proposes a set of eight methods to combine using M-Coffee, but also says that all fifteen can be included as long as you apply a weighting to each method.

According to Pais et al. [144], consistency-based multiple sequence alignment programs, like T-Coffee, generally produce better results than other types of programs. Unfortunately, at time of writing, we could not find any independent (i.e. not by the authors) review paper which included M-Coffee in the comparison.

M-Coffee was chosen as the alignment program because it includes information from several methods.

Even after alignment, it was necessary to consider whether all positions in the alignment had been aligned correctly. For this, the TCS (Transitive Consistency Score) [27] was used; this is built-in to T-Coffee and therefore easy to use in conjunction with that program. This was chosen because TCS 'manages to bring all [alignment] methods at a comparable level of accuracy' [27, pp. 1627]. The Filtered TCS method was chosen, which uses the ColumnTCS measure to filter out columns less than some pre-specified threshold. Compared to alternative methods such as GUIDANCE (Guide tree based Alignment Confidence) and HoT (Heads or Tails), TCS was found to be better at identifying correctly aligned characters in an alignment. The TCS column threshold of 4 was used as this is the default used in the TCS web server [26].

4.2.2 Data collected

Now that we have described the methodology used to collect the data, we shall briefly describe the nature of the data collected. 17479 alignments were constructed, with the following characteristics:

- Minimum number of sequences = 2
- Maximum number of sequences = 80
- Mean number of sequences ≈ 6
- Median number of sequences = 6
- Mode number of sequences = 5
- Minimum alignment length = 51
- Maximum alignment length = 8792
- Mean alignment length ≈ 574
- Median alignment length = 429
- Mode alignment length = 312

4.3 Summary

In this chapter we compared several databases for protein sequences, and described how we constructed our data set of "real" data using our chosen database, HomoloGene, and also how we created alignments from these sequences. In the next chapter we derive parameters from this data set for use in our simulations, and conduct the simulations themselves.

Chapter 5

Simulation

In the previous chapter, we collected real data from HomoloGene. We shall now use it to derive parameters for our simulation, such that the resultant simulated data is as close as possible to the real data. As detailed earlier, the correlation signal observed between pairs/groups of sites is made up of several components, some of which we are less interested in (for example, the correlation due to phylogenetic background, $C_{phylogeny}$, and the correlation due to randomness, $C_{stochastic}$). Our aim was to test our algorithms according to how well they could reduce the effect of these two background signals, while detecting (what appears to be) the "true" co-evolution signal. "Real" data would not be effective for this purpose because most of the time we do not know for certain which effects can be attributed to which signals. Also, we very rarely know for certain where the "real" co-evolving sites are. To overcome this, we decided to generate simulated data so that we know for certain where the co-evolving sites are in our data.

This chapter first details the simulation program chosen, and then the process of deriving the necessary parameters for the simulation. The steps this chapter covers are shown in Figure 5.1.

5.1 Simulation program

In this section we compare several programs for simulating protein sequences, and give the reasons for our choice of program.

We identified several papers where the authors developed their own method of simulating alignments with co-evolving sites [181, 65, 116, 56], but we did not use these due to the reasons outlined below:

- Tillier and Lui [181] – relatively small number of co-evolving sites used (a constant number, 10, was used for all alignment sizes (40 - 320 rows

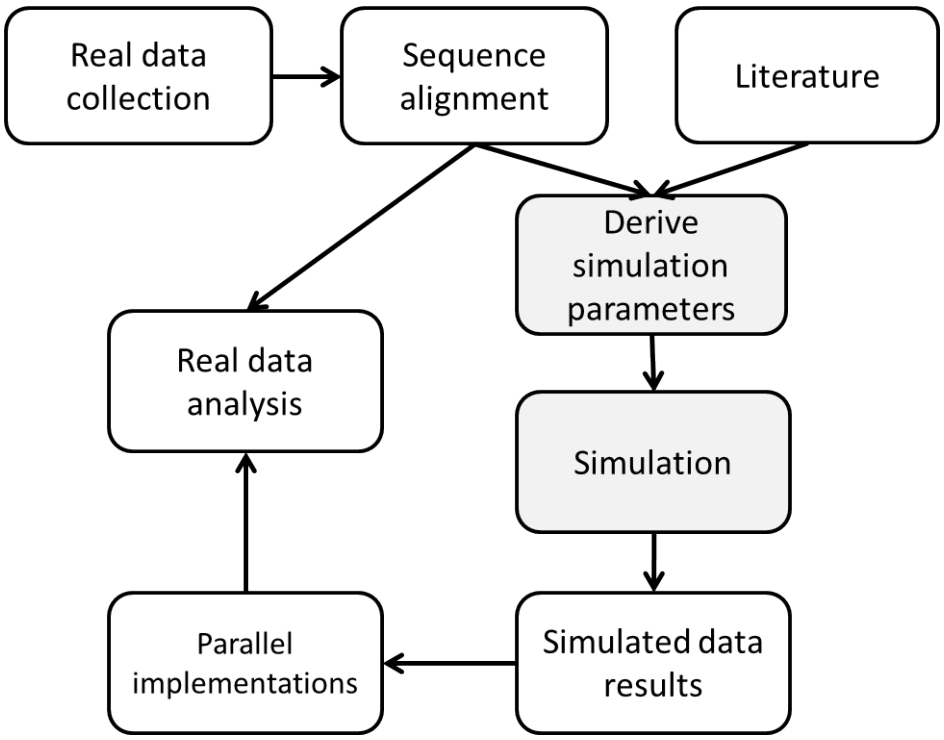

Figure 5.1: This chapter covers the derivation of simulation parameters from a combination of information from the real data collected in the previous chapter, and information from the literature, as well as the simulation process itself (shaded boxes).

and 50 - 500 columns)), and we wanted to vary this across different alignment sizes

- Fodor and Aldrich [65] – the alignments it produces are only 2 columns wide, whereas we want to use both large and small alignments, to see what effect this has on the results obtained
- Martin et al. [116] – like the method of Tillier and Lui, it used a constant number of co-evolving sites, and it also used a constant number of rows and columns in the alignment
- Fares and Travers [56] – same reason as for Martin et al. [116]

We then decided to investigate several programs implemented solely for the purpose of simulating molecular co-evolution (as opposed to being developed for the authors of a paper to use once for their study and therefore possibly tied to the kind of data they were trying to simulate). We began by visiting the NIH's Genetic Simulation Resources website [126] and used their comparison tool to filter out programs which simulated protein sequences; this gave us ALF [35], CASS [79], CS-PSeq-Gen [183], EvolveAGene [85], FLUX SIMULATOR [82], indel-Seq-Gen [174], Indelible [62], ProteinEvolver [8],

Rose [173], Seq-Gen [159], and Simprot [145]. We had also already, independently found out about the program MSAvolve [1], so we added this to the list to compare. Out of all of these, only CS-PSeq-Gen and MSAvolve actually simulate molecular co-evolution. Of these two, we decided on CS-PSeq-Gen [183] as we were encouraged by the fact that it was based on PSeq-Gen [80], a program which we understand to be widely-used in the simulation of protein sequences. PSeq-Gen simply evolves protein sequences according to a given phylogenetic tree input, but CS-PSeq-Gen expands on this by allowing for the random generation, or user specification, of clusters of correlated sites in the protein sequence.

Having chosen our simulation program, we now need to choose the values that each of the CS-PSeq-Gen parameters shall take in our simulations.

5.2 Simulation parameters

We shall now look at the CS-PSeq-Gen simulation parameters, describing each and detailing how we chose the values of each parameter to use. To work out biologically plausible values for these parameters, we consulted various studies which had used the PSeq-Gen program. In cases where consulting the literature did not provide conclusive suggestions, we did experiments to work out the values to use. We also, of course, looked at the data collected in the previous chapter.

5.2.1 Evolution model

PAM, JTT (Jones-Taylor-Thornton, after the authors) and MTREV (Mitochondrial Reversible) are the evolution models offered by the program. PAM was described in Section 2.2.2. The method for constructing the JTT similarity matrix is very similar to that used to construct PAM, the main difference being some approximations are made in order to speed up the construction itself [97]. MTREV [2] used proteins encoded in mitochondrial DNA (as opposed to nuclear DNA) to construct the substitution matrix using a maximum likelihood method. All but one of the papers which used PSeq-Gen that we consulted chose to only use JTT [5, 125, 198, 93, 87], so we only use JTT too.

5.2.2 Alignment dimensions

The size of the alignment produced is affected by two different parameters: the alignment length and the tree (the number of leaves in the tree specifies the number of sequences in the alignment).

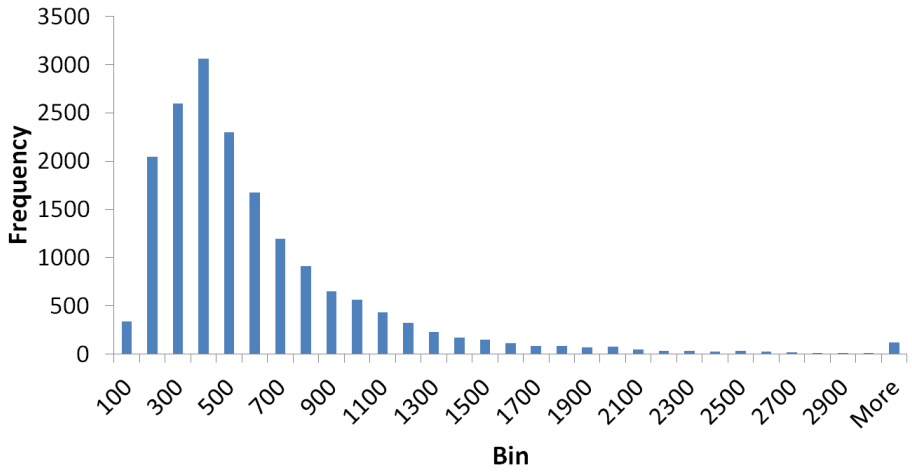

Figure 5.2: A histogram showing the frequency of different "real data" alignment lengths in terms of number of amino acids. Bin size is 100.

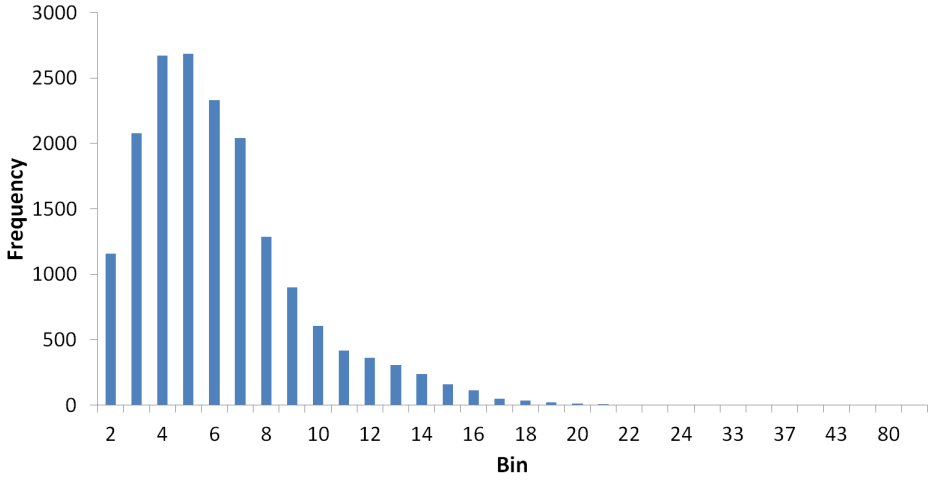

Figure 5.3: A histogram showing the frequency of different "real data" cluster sizes (i.e. number of sequences in the alignment). Bin sizes are unique, so this is not technically a histogram, but this was the easiest way to show the various sizes.

Wanting our programs to be usable for "real" data in the future, we looked toward the real data we collected for ideas. Details of the methods for obtaining this data can be found in Chapter 4.

We constructed histograms for the lengths of the HomoloGene alignments we had constructed (Figure 5.2), and for the number of sequences in those alignments (Figure 5.3). For the alignment length histogram, we can see that the real data alignments tend to be fairly short; using this alignment for guidance, we chose to simulate alignments with lengths 50, 600, and 1200. Also, for the number of sequences histogram, the values tend to also be small in real data. We could have chosen 2 as a number of sequences to be simulated, but we felt this would not give enough information, so we began at 6 sequences, and also simulated alignments with 10 and 22 sequences.

CS-PSeq-Gen requires a tree, so that it can simulate the evolution of sequences down the tree (this is in Newick format, described in Section 2.4.1). For each of the real data alignments, we constructed a phylogenetic tree using ClustalW2 (described in Section 2.4.2). We used the following command:

```
clustalw2 -infile="${words[0]}_dir/
${words[0]}${words[1]}.txt.fa" -tree
-clustering=upgma
```

From the set of trees constructed from the real data, we chose one with 6 leaves, one with 10, and another with 22. These would be the trees we would pass to CS-PSeq-Gen for our simulations.

The branch lengths of the trees constructed are measured as 'the number of substitutions as a proportion of the length of the alignment (excluding gaps)' [50]. However, CS-PSeq-Gen assumes branch lengths are equal to 'the number of substitutions observed along branches' [81], i.e. it is an absolute value, whereas ClustalW2 gives branch lengths as a value relative to the length of the alignment we gave it. This means that before we can run our simulations, we must work out the appropriate branch scale parameters to use.

5.2.3 Branch scale

Given the tree we are using to generate our data, we can apply a branch scaling factor to this tree. This factor is multiplied with each branch length to get the new, scaled, branch length, which serves to increase the branch lengths if the factor is > 1, or decrease the branch lengths if the factor is < 1.

We needed to work out a branch scale factor for each combination of tree (number of sequences) and alignment length. We did so by comparing the Percentage Identity Matrix (PIM) obtained for each alignment from which we derived the tree against the matrix obtained for an alignment simulated using that tree (and a particular alignment length). (The Percentage Identity Matrix for an alignment is a matrix where each value denotes, for a different pair of sequences within that alignment, the percentage of characters within those two sequences that are equal. An example is given in Figure 5.4.) The general format of our comparison method is given below:

1. Given a set of simulated alignments generated from a particular tree, ensure these are in FASTA alignment format (CS-PSeq-Gen generates alignments in PHYLIP format, so conversion is usually necessary)

	0	1	2	3	4	5
0	G	G	T	G	C	C
1	C	A	C	T	T	T
2	A	G	G	T	G	C
3	T	C	C	A	A	T

-	0	0.33	0
0	-	0.17	0.33
0.33	0.17	-	0
0	0.33	0	-

Figure 5.4: An example showing how a Percentage Identity Matrix (PIM) can be constructed from an alignment. For example, the value in the third cell in the first row is 0.33 because the sequences in rows 0 and 2 have two characters identical (at positions 1 and 5) (2/6 = 1/3 = 0.33).

2. Use a program such as ClustalW2 to obtain the Percentage Identity Matrix (PIM) for each alignment, and for the original alignment from which the tree for simulation was generated

3. Use R[1] to perform a Wilcoxon rank sum test to compare each simulated alignment's PIM with the original alignment's PIM

4. Average p-values over length, or branch scale, whichever parameter we are interested in measuring the effect of

We iterate over the above process for different values of the branch scale factor, and choose the one which yields the maximum average p-value. In the Wilcoxon rank sum test performed, the null hypothesis we are testing is that the two groups being compared (the two Percentage Identity Matrices) are the same, and the alternative hypothesis is that these groups are different. A p-value less than some threshold (usually 0.05) tells us that we can reject the null hypothesis (and so we have reason to believe the two groups are different). Here, we reverse this definition, and hypothesise that the higher the p-value, the more likely the two groups (in our case, the PIM values for the simulated alignment and the real alignment) are to be similar. This is why we are looking for large p-values. (It should be noted that the p-value does not

[1]R is a programming language for performing statistical analyses [158]

actually denote the strength of the relationship, but for the sake of obtaining good values of the branch scale parameter, we ignore this.)

Completing this process, we obtained the following branch scale factors for each combination of alignment length (l) and number of sequences (s):

- $s = 6, l = 50$: 43
- $s = 6, l = 600$: 460
- $s = 6, l = 1200$: 940
- $s = 10, l = 50$: 29
- $s = 10, l = 600$: 390
- $s = 10, l = 1200$: 720
- $s = 22, l = 50$: 35
- $s = 22, l = 600$: 370
- $s = 22, l = 1200$: 750

5.2.4 Gamma scale parameter

This parameter influences the amount of mutation rate variation between positions in the alignment generated – a value of α less than or equal to 1 produces a distribution that is L-shaped (so most positions have a low mutation rate, and a small number have a relatively high mutation rate), whereas a value greater than 1 produces a distribution which is more Normal-like (so most positions have some median mutation rate, with smaller numbers of positions having particularly high or low mutation rates) [201]. (Figure 5.5 shows how this distribution looks for different values of α.)

We varied the value of this parameter as we cannot say whether rates differ significantly or not in real life – 'substitution rate variation exists among sites in almost all genes or proteins, with the possible exception of some pseudogenes or "junk" DNA' [201, pp. 367]. We looked towards the literature for suggestions as to possible values to use:

- [125] used $\alpha = 1$ and did not say why
- [87] used $\alpha = 1$ and 100 to model among site rate variation
- [198] used $\alpha = 1$, suitable for the data they were going to use
- [41] used $\alpha = 0.7$ and did not say why

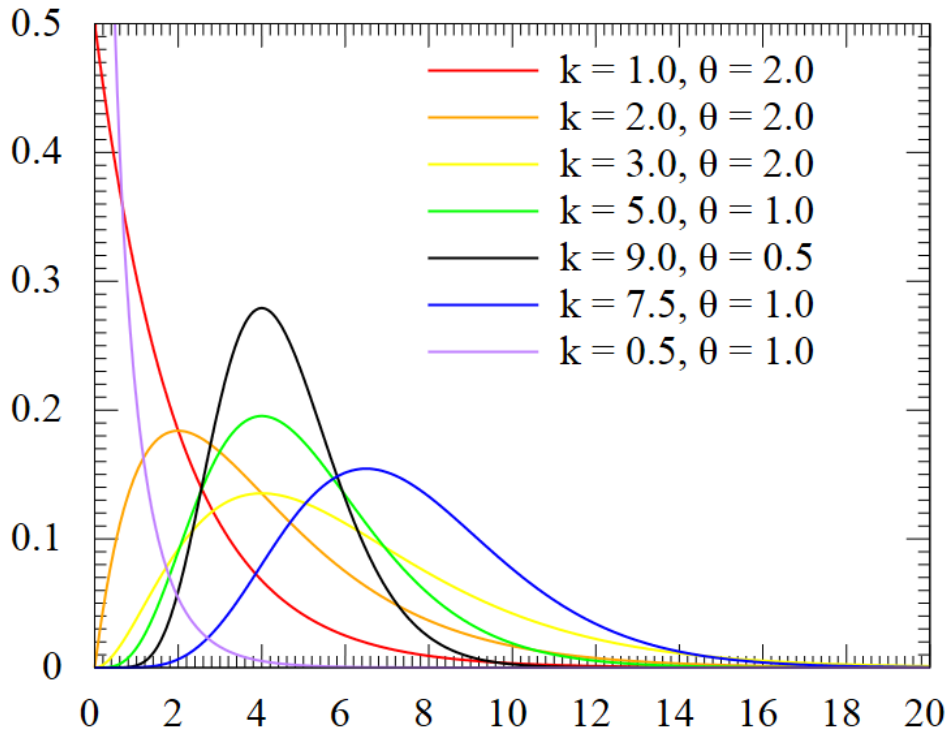

Figure 5.5: The probability density function of the Gamma distribution, showing the distribution for different values of k (α) and θ (β); see Section 2.2.1 for more details. Figure obtained from [11] (available under Creative Commons license; see citation for details).

Considering we wanted to vary the amount of mutation rate variation, and also taking into account how the distribution changes shape depending on the value of α, we chose three values for our simulations: 1, 100, and no value (which means there is no rate variation between sites).

5.2.5 Co-evolving clusters

As far as the size of co-evolving clusters and number of clusters is concerned, most of the methods used in the papers consulted simply detect pairs [1, 17, 20, 22, 28, 29, 30, 32, 40, 42, 44, 45, 56, 58, 65, 68, 69, 95, 114, 131, 143, 153, 168, 181, 193, 195]. A good selection of other papers also detect larger groups of co-evolving sites; the values used by these are given in Appendix A.2.

To decide which values to use, we picked a minimum, "average", and maximum value of the number of clusters and sites per cluster to use based on information from the literature. We chose to have 1, 8, and 44 clusters, and then 2, 13, and 67 sites per cluster (as well as an extra, random, number of sites per cluster which means that the simulation program chooses the number of sites per cluster in between 2 and 67 randomly; in this case, the number of

sites per cluster can differ between clusters).

5.2.6 Amino acid frequencies

We also needed to consider how the sequence at the root of the tree is generated by the simulation program. We did so by looking at the code for the simulation program itself, since the original paper [183] did not give the detail we needed. The program has an array, $freq$, whose values are amino acid frequencies according to Dayhoff [36] divided by the sum of all of the frequencies. It then has an array, $addFreq$, calculated as follows:

- $addFreq[0] = freq[0]$
- $addFreq[i] = addFreq[i-1] + freq[i]$

Each amino acid of the root sequence is chosen by generating a random number between 0 and 1, starting at the beginning of $addFreq$, and incrementing the index into $addFreq$ as long as the value of $addFreq[i]$ is less than the random number generated. Wherever we stop incrementing (when $addFreq[i]$ is at least as big as the random number), that index tells us which amino acid to choose.

In the end we decided to simply use the default Dayhoff frequencies as they are programmed into CS-PSeq-Gen because those are taken from a large amount of real data ('71 groups of closely related proteins' [36, pp. 345]). (In reality, CS-PSeq-Gen has made a small change to the frequencies being used – they have changed the frequency of K from 0.081 to 0.080 in order to achieve a total sum of exactly 1, rather than 1.001, but this is only a small change.)

5.3 Iterations

The parameters chosen are:

- Evolution model: JTT
- Alignment length: 50, 600, 1200
- Number of sequences per alignment: 6, 10, 22
- Gamma scale parameter: none, 1, 100
- Number of co-evolving clusters: 1, 8, 44
- Co-evolving sites per cluster: 2, 13, 67, random (between 2 and 67)

We were able to collect results for 100 iterations per set of parameters; by this we mean, we simulated one hundred alignments for each set of parameters, and generated results for each method, for every single alignment generated. As a more general comment on the data generated that we used for collecting results, CS-PSeq-Gen can actually produce "co-evolving sites" which are in reality, fully conserved, i.e. not co-evolving at all. Because of this, we needed to filter through the results in order to only count those whose co-evolving sites are actually co-evolving (not conserved). This meant that some alignments' results were not able to be used. We used a total of 5510 alignments in our analysis.

5.4 Summary

In this chapter we first described how we chose our simulation program, CS-PSeq-Gen, from the possibilities available to us. Simulated data is preferable over real data for comparing the co-evolution detection methods as we can say for certain where the co-evolving sites are in simulated data, and we can also constrain the parameters of the simulation such that the resulting data emulates the real data. The simulation parameters were also chosen in this chapter. The evolution model, gamma scale parameters, and co-evolving clusters were chosen using information from the literature. The alignment dimensions were chosen looking at the dimensions of the real data alignments constructed in the previous chapter. The branch scale parameters were chosen using experimentation. Now that we have simulated the data necessary to compare the co-evolution detection methods, we can move forward with the comparison itself, detailed in the next chapter.

Chapter 6

Comparison of Serial Methods

At the end of Chapter 3 we chose the five methods that we shall be comparing in the current chapter. In Chapter 4 we collected real data (the box labelled "Real data collection" in Figure 6.1), which was used to inform the parameters for the simulation, which we described in Chapter 5, as well as the rationale for these choices (the boxes labelled "Derive simulation parameters" and "Simulation" in Figure 6.1). It is this simulated data that we shall be using to evaluate each of the co-evolution detection methods in the comparison.

This chapter describes the results of the comparison of five methods for detecting molecular co-evolution (CAPS, PlotCorr, Mutual Information, Waddell – Mutual Information, and Waddell – Kappa), using the simulated data generated in the previous chapter. We then make recommendations as to which method(s) to parallelise. The steps this chapter covers are shown in Figure 6.1. We begin by giving an overview of the general process of gathering the results, and show the results themselves. At the end of this chapter we draw together our findings for the various methods considered in order to identify the most suitable methods for the parallelisation described in the next chapter.

6.1 Results-gathering approach

First we shall outline the approach used to gather the results from the simulated data. As discussed before, we used simulated data because with real data we cannot be sure where the truly co-evolving sites are; however we can control this in simulated data, allowing for a fairer comparison.

As discussed in Chapter 3, we decided to compare two approaches to calculating Z-scores for our results. The aim of Z-scores is to remove the effect

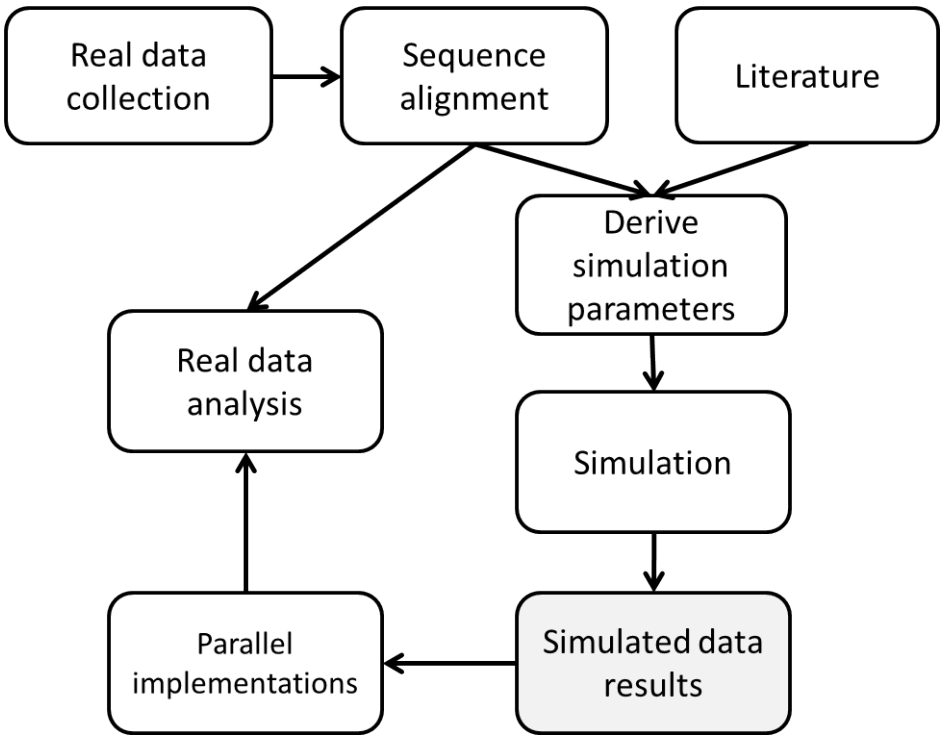

Figure 6.1: This chapter covers the comparison of the co-evolution detection methods on simulated data (shaded box).

of background noise from the correlation values. As a reminder, the Z-score equation is given by:

$$Z = \frac{x' - \overline{x}}{\sigma(x)} \tag{6.1}$$

where x' is a correlation value for a pair of columns in the input alignment. $\overline{x}$ is the mean, and $\sigma(x)$ the standard deviation, of either (i) the correlation values for all pairs of columns (for the original values Z-score), or (ii) the correlation values for this pair of columns after shuffling the characters within the columns in place (for the shuffled Z-score). Once we have generated a Z-score for each pair of positions (according to a particular method such as CAPS, PlotCorr, etc.) we need to choose a *Z-score threshold*. The pairs of co-evolving sites generated by each method consist of those pairs with a Z-score greater than some threshold. We chose Z-score thresholds of 0–3 inclusive (integers only) after looking at the Z-scores we obtained and the statistics for those results for a small amount of data (we describe the statistics below). Note that here we are talking about the classification of the sites, which may be different to the ground truth of whether that pair of sites is co-evolving or not. The following table illustrates this:

	Classified as co-evolving	**Classified as not co-evolving**
Truly co-evolving	True positive (tp)	False negative (fn)
Truly not co-evolving	False positive (fp)	True negative (tn)

For each of the Z-scores generated for each pair of sites (and given a chosen Z-score threshold), we can categorise that Z-score as a true positive, true negative, false positive, or false negative, according to whether that pair is actually co-evolving and whether it was classified as such.

There are statistics we can use to summarise the counts of true positives, true negatives, etc. that we have for a particular combination of Z-score type, threshold, method, and input alignment. The ones we decided to focus on are:

- Sensitivity = $\frac{tp}{tp+fn}$; i.e. that method's ability to classify truly co-evolving sites as such
- Specificity = $\frac{tn}{tn+fp}$; i.e. that method's ability to classify truly not co-evolving sites as such
- Precision = $\frac{tp}{tp+fp}$; i.e. given that a pair of sites has been classed as co-evolving, how likely is it that they are actually co-evolving?

For each combination of Z-score type (shuffle and original values), threshold (0, 1, 2, 3), method (CAPS, PlotCorr, MI, Waddell – MI and Waddell – Kappa), and alignment, we calculated the sensitivity, specificity, and precision. It should be noted that the simulation program used, CS-PSeq-Gen, can define certain sites as co-evolving when these may not be detected as such by the detection methods. For a pair of sites to be classified as co-evolving by the detection methods, they need to have accumulated mutations during sequence evolution – the more mutations that have occurred, the easier co-evolution is to detect. Conserved sites will never be detected as co-evolving, so we only consider those sites defined as co-evolving by CS-PSeq-Gen that are also not conserved.

A diagram showing the process we shall be following within this chapter is shown in Figure 6.2. We shall begin by establishing which of the two approaches to calculating Z-scores gives us better performance, and then "fixing" this such that we only present results using this approach ('Choose Z-score type (original values or shuffle)'). As mentioned above, conserved sites

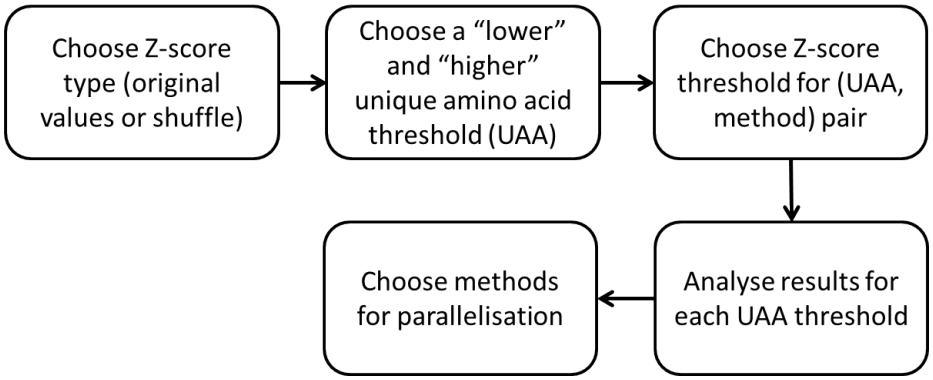

Figure 6.2: The process of analysis this chapter uses.

cannot be classified as co-evolving, but certain methods may give better performance for higher (more unique amino acids) or lower (fewer unique amino acids) diversity within a column of the alignment. To investigate this, we choose a "lower" and a "higher" unique amino acid threshold and fix these for the rest of the chapter ("Choose a 'lower' and 'higher' unique amino acid threshold (UAA)"). The choice of Z-score threshold also influences the performance of the methods compared, so for each combination of unique amino acid threshold and co-evolution detection method, the Z-score threshold yielding the best performance is chosen ("Choose Z-score threshold for (UAA, method) pair"). These three steps are undertaken to ease the presentation of the results. We can then analyse the results while we vary other parameters, and finally use this analysis to inform the choice of co-evolution detection methods for parallelisation ("Choose methods for parallelisation").

In the next section we shall "fix" the Z-score type, and the Z-score threshold(s), so that we do not have to show all 4 Z-score thresholds for both Z-score types.

6.2 Choosing a Z-score type

As was mentioned above, we want to "fix" the values of some parameters so that we do not need to show all of the results for all of the parameters, to make it easier to present the results. First we shall choose which is the better-performing Z-score approach, original values or shuffle. Once we have established which is the best of the two, we shall only present the results gathered using that Z-score approach for the rest of this chapter.

As a reminder, the original values Z-score approach uses the mean and standard deviation of the correlation values for all pairs in the alignment in the calculations. In contrast, the shuffle Z-score approach shuffles the two

Statistic	Wilcoxon	Sign
Sensitivities	1	6.087e-05
Specificities	<2.2e-16	<2.2e-16
Precisions	1.552e-14	0.9034

Table 6.1: p-values for the Wilcoxon sign-ranks test (the "Wilcoxon" column) and the sign test (the "Sign" column) for comparisons between the sensitivities, specificities, and precisions, comparing the results from the two Z-score types. We used alternative='greater', showing that we believe that the original values Z-score type would outperform the shuffle Z-score type. The Wilcoxon sign-ranks test was repeated for sensitivities with alternative="less", and obtained the p-value of 2.2e-16. The sign test was repeated for precisions with alternative='less', and obtained the p-value of 0.1954. Further explanation is given in the main text.

Statistic (mean)	Original values Z-score	Shuffle Z-score
Sensitivities	0.7444162	0.7809325
Specificities	0.9151887	0.6808495
Precisions	0.07606378	0.07018549

Table 6.2: Average sensitivities, specificities, and precisions for the two Z-score types.

columns in a pair in place, mixing up the characters within each column, before re-calculating the correlation for that pair. This is repeated some number of times, and then the mean and standard deviation of these correlation values after shuffling are used in the Z-score calculations. (See Appendix A.1 for an example using each of these Z-score approaches.)

To work out which is the better approach to calculating Z-scores, we shall compare the sensitivities, specificities, and precisions obtained for all of the alignments, for each of the two Z-score types.

We want to compare the set of original values Z-score sensitivities to the set of shuffle Z-score sensitivities, and the same for the specificities and precisions, to establish which Z-score type gives us the better results. The sensitivities, specificities, and precisions sets are all non-Normal, meaning we must use a non-parametric test. Our data comfortably satisfies the assumptions of the Wilcoxon signed-ranks test, except for one assumption, that is, the differences between the two groups (sensitivities for the original values Z-score, sensitivities for the shuffle Z-score, for example) should be symmetrical. If this were true, then we could use the Wilcoxon signed-ranks test; if not, we would need to use the sign test. Because it is inconclusive whether the differences are symmetrical or not, we have calculated the p-values for both tests. (More detail is given in Appendix A.3.)

V	F	L	V	P	E
I	D	M	M	E	V
V	G	A	I	F	C
I	L	A	L	N	K
K	W	D	S	A	E

Figure 6.3: An alignment with two co-evolving groups (each with two sites) highlighted in different colours. (The characters within this toy alignment are quite random simply to illustrate the different unique amino acid thresholds without showing too many sequences.)

The Z-score methods differed significantly for 5 out of a total of 6 tests performed (2-sided tests), with the original values Z-score outperforming the shuffle Z-score on 4 of the comparisons (see Table 6.1). We therefore chose to use the original values Z-score.

Now that we can "fix" the Z-score type to the original values Z-score type (we only consider this Z-score type's results from now on), we shall look at the overall effect that the amount of conservation in a column has on each detection method's results.

6.3 Analysing results per UAA threshold

First it should be noted that it became apparent that CS-PSeq-Gen can produce groups of positions which are defined by the program as co-evolving, but which are not identifiable as such, due to at least one of the columns within the group having only one unique amino acid. Identifiable co-evolving sites are those with at least 2 unique amino acids (UAA) within the alignment column. Because of this, when calculating the sensitivities, specificities, and precisions, we effectively re-classified those "co-evolving" sites that were actually fully conserved as not co-evolving.

Each co-evolution detection method could be influenced by the number of amino acids in the columns being examined, so in this section we establish, for each method, whether that method performs better with a higher or lower diversity of amino acids in the alignment columns (this relates to the idea of conservation, which we discuss in more detail now).

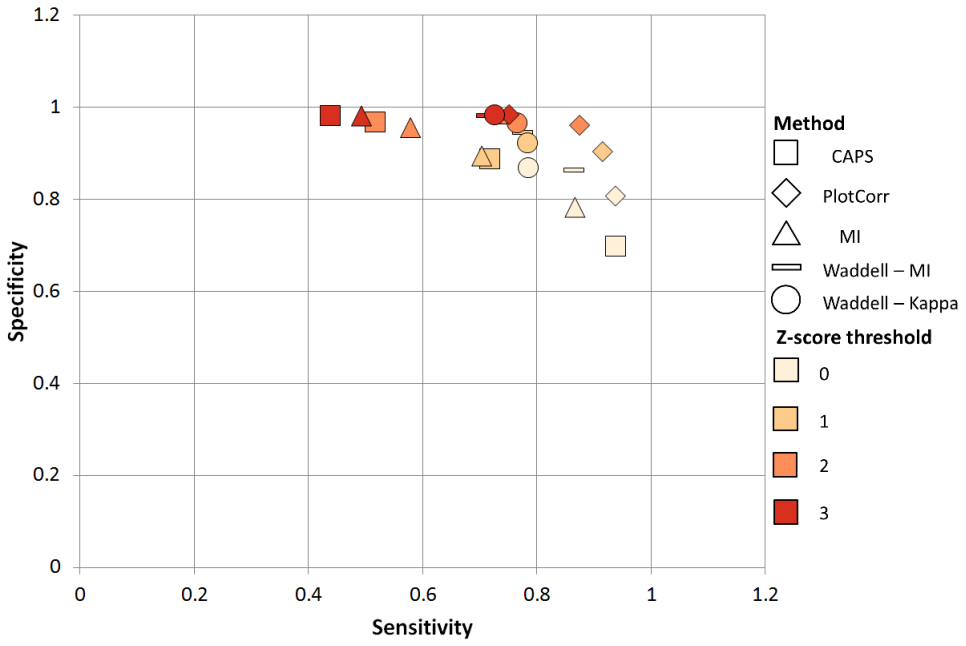

Figure 6.4: Mean sensitivities and specificities for the original values Z-score, for unique amino acid threshold 2. Different methods are indicated by the differently shaped symbols, and different Z-score thresholds are indicated by the different symbol colours, as indicated in the legend.

We wanted to investigate what influence conservation has on the results, so we introduce the concept of a unique amino acid threshold. This is a threshold number of unique amino acids in a column that we set such that any sites with less than that number of unique amino acids are classed as non-co-evolving during the analysis. For example, within the alignment given in Figure 6.3 there are two co-evolving clusters, indicated by the two groups of different-coloured columns. Setting the threshold number of unique amino acids to 3, both pairs of columns will be considered co-evolving; the first, second, fourth, and fifth columns all have at least 3 unique amino acids. However, setting the threshold to 5, only the second group (consisting of the fourth and fifth columns) would be classed as truly co-evolving. The greater the number of unique amino acids within a column, the more information that is available for identifying co-evolving sites.

We shall now analyse results while varying the unique amino acid threshold. We only consider results from the original values Z-score type, having established in the previous section that this is the better-performing of the two Z-score methods.

It is worth making the point that for all co-evolution detection methods, the following is true: as the Z-score threshold increases, specificity increases (a higher threshold means that sites that are not co-evolving would be more

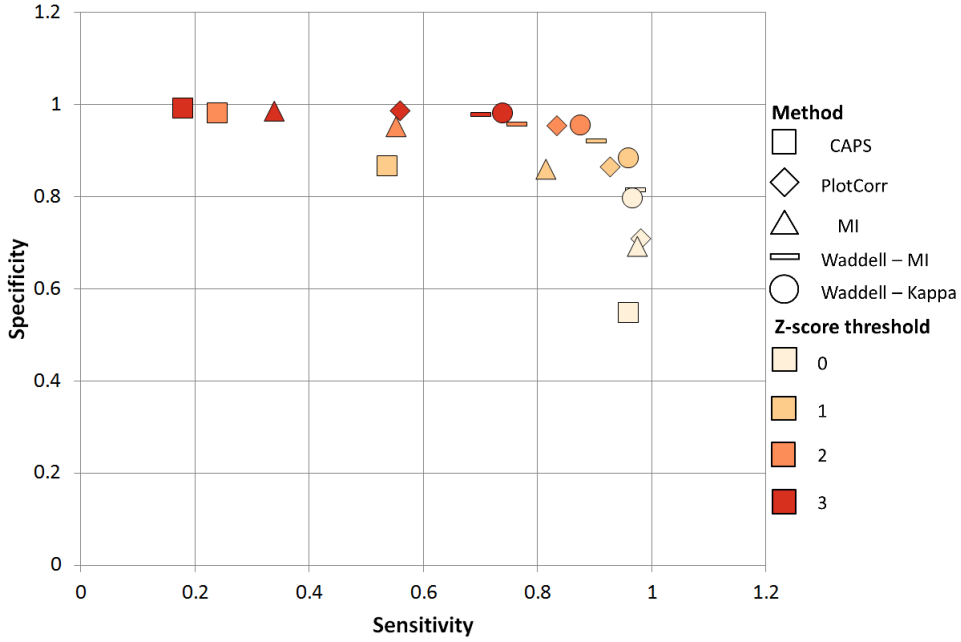

Figure 6.5: Mean sensitivities and specificities for the original values Z-score, for unique amino acid threshold 3. Different methods are indicated by the differently shaped symbols, and different Z-score thresholds are indicated by the different symbol colours, as indicated in the legend.

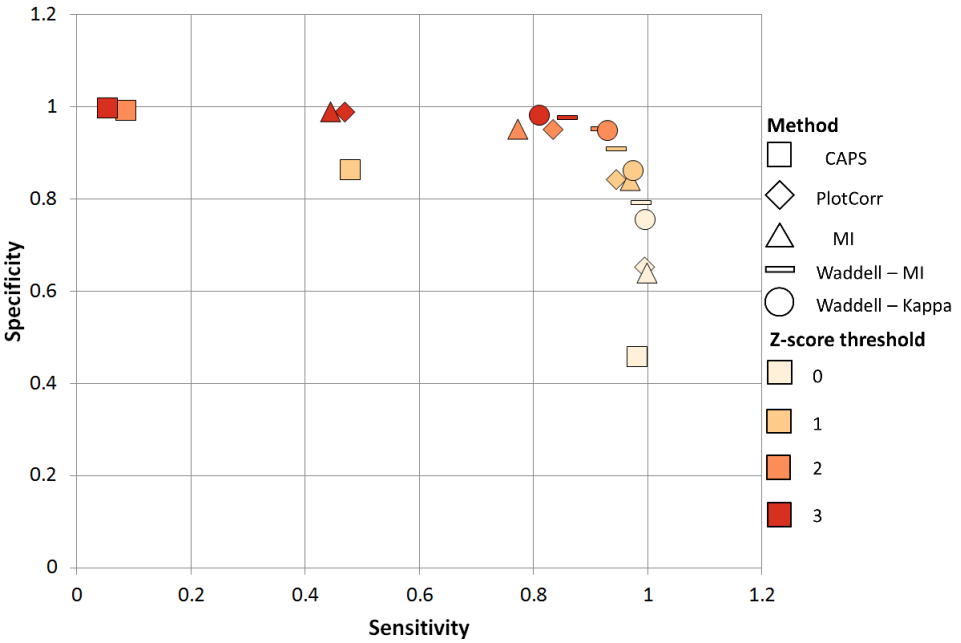

Figure 6.6: Mean sensitivities and specificities for the original values Z-score, for unique amino acid threshold 4. Different methods are indicated by the differently shaped symbols, and different Z-score thresholds are indicated by the different symbol colours, as indicated in the legend.

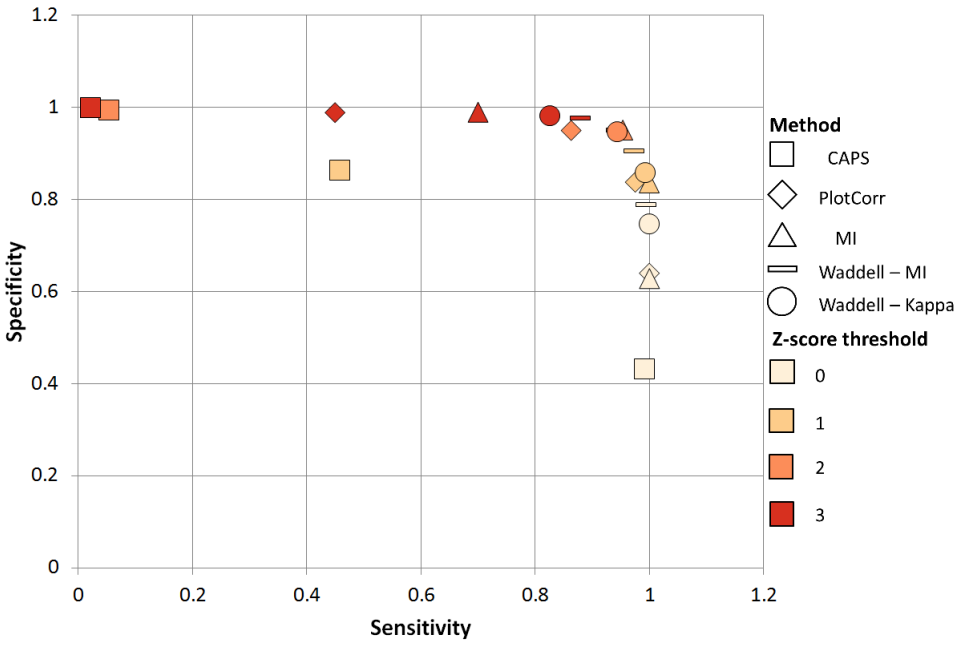

Figure 6.7: Mean sensitivities and specificities for the original values Z-score, for unique amino acid threshold 5. Different methods are indicated by the differently shaped symbols, and different Z-score thresholds are indicated by the different symbol colours, as indicated in the legend.

likely to be classed as such) but sensitivity decreases (as a stricter criterion will inevitably mean some true positives are missed), as is to be expected.

First it shall be investigated how different methods perform with varying unique amino acid thresholds. For each method, and for each Z-score threshold, the best-performing UAA threshold was chosen. (See Figure 6.4, Figure 6.5, Figure 6.6, and Figure 6.7 for the average results for each UAA threshold.) CAPS and PlotCorr always perform best with the lowest unique amino acid threshold (2), whereas the other three methods vary depending on the Z-score threshold chosen. For these methods, the best UAA threshold tends to increase as the Z-score threshold chosen increases, although in many cases it can be seen that results are very close between thresholds. If we examine all points available for sensitivity, specificity, and precision (the latter not plotted on the graphs for simplicity), we can again see that certain methods perform best with certain UAA thresholds. The UAA thresholds which allow the best performance for each co-evolution detection method were established using the Wilcoxon rank-sum test. These trends are summarised (see Appendix A.4 for the full list of p-values):

- CAPS – to improve the precisions and sensitivities, a lower UAA is better, for specificities a higher UAA is better
- PlotCorr – for precisions and specificities a lower UAA is better, for

sensitivities there is no such pattern

- MI – for precisions and specificities a lower UAA is better, for sensitivities a higher UAA is better
- Waddell – Kappa/Waddell – MI – for precisions and specificities a lower UAA is better, for sensitivities a higher UAA is better

Overall, for all the methods but CAPS, we obtain better results with less diversity per column (a lower UAA threshold) for precisions and specificities, and more diversity improves the sensitivities. This makes sense intuitively, because Mutual Information for example tends to give higher scores to positions with a higher entropy (which translates to a higher number of unique amino acids). This means that having a higher number of unique amino acids increases the chance of detecting a pair accidentally, whereas lowering the number of unique amino acids increases the likelihood of categorising the true negatives correctly. In the case of CAPS, however, the opposite is true for sensitivities and specificities.

We can see that some methods perform better with a lower unique amino acid threshold, whereas others perform better with a higher unique amino acid threshold. To account for this, and to make it easier to look at individual parameters such as alignment size without having to also consider all unique amino acid thresholds, we shall choose a "lower" and a "higher" unique amino acid threshold.

6.4 Choosing a "lower" and a "higher" unique amino acid threshold

To choose an appropriate "lower" and "higher" unique amino acid threshold, we shall look at the number of co-evolving blocks ("clusters"/"groups") that each unique amino acid threshold gives us to consider, so that we can choose thresholds that give us enough information. The "lower" threshold will be either 2 or 3, and the "higher" threshold will be either 4 or 5. As the unique amino acid threshold is increased, there are inevitably fewer co-evolving groups we can look at, but we want to find out how large this decrease is. If the drop between the number of blocks for threshold 2 and threshold 3 is too large, for example, we would choose threshold 2 as the "lower" threshold, even though a threshold of 3 gives us more information to work with (through a greater diversity of amino acid characters).

Method	Best Z-score threshold	Sensitivity	Specificity
CAPS	1	0.716941	0.887772
PlotCorr	2	0.874779	0.960892
MI	1	0.703377	0.894003
Waddell – MI	0	0.865331	0.863097
Waddell – Kappa	3	0.726281	0.983680

Table 6.3: The mean sensitivity and specificity values for the best Z-score chosen for each method, for unique amino acid threshold 2.

Method	Best Z-score threshold	Mean sensitivity	Mean specificity
CAPS	1	0.478902	0.862558
PlotCorr	2	0.834128	0.950833
MI	1	0.969237	0.840161
Waddell – MI	3	0.859651	0.977096
Waddell – Kappa	3	0.810084	0.982038

Table 6.4: The mean sensitivity and specificity values for the best Z-score chosen for each method, for unique amino acid threshold 4.

The number of co-evolving groups for each threshold are:

- 2: 20640 (from 5510 alignments)
- 3: 5893 (from 1938 alignments)
- 4: 1589 (from 759 alignments)
- 5: 385 (from 273 alignments)

For the "higher" threshold, we shall just choose 4 because of the fairly large difference (between 4 and 5) in the number of blocks this gives us to analyse. For the "lower" threshold, it is not so obvious which threshold to choose as there are a lot of co-evolving blocks for both 2 and 3, so we consider that the best methods for "lower" thresholds are CAPS and PlotCorr, and compare the results for these thresholds to see where they perform better. (See Appendix A.5 for this comparison.) We have chosen threshold 2 as our "lower" threshold.

6.5 Choosing Z-score thresholds

Having now "fixed" the definitions of "lower" and "higher" unique amino acid thresholds, we must choose the best Z-score threshold for each combination of unique amino acid threshold and co-evolution detection method. We do so to, again, make the presentation of the results easier, as showing

the results for all 4 Z-score thresholds (0–3) would become confusing when we start considering all the different values of other parameters too, such as alignment length.

The best Z-score thresholds are chosen by inspecting the graphs within Figures 6.4 and 6.6; starting from the cell on the graph's grid which encompasses those points with both sensitivity and specificity $>= 0.8$, look for points for the co-evolution detection method of interest. If there are no points for this method, inspect the cell to the left of this one; repeat until there is at least one point for this method in the cell we are inspecting. For example, for UAA threshold 2, for CAPS, there is no square-shaped point within the sensitivity and specificity $>= 0.8$ cell, so we look to the left, and find a point for Z-score threshold 1 within the cell for sensitivity between 0.6 and 0.8, specificity $>= 0.8$. This is the chosen Z-score for CAPS here. If there are multiple points within the cell we are inspecting, as in the case for Waddell – Kappa for UAA threshold 2 (4 points within the cell for sensitivity between 0.6 and 0.8, specificity $>= 0.8$), we prefer a higher specificity over sensitivity (so we would choose Z-score threshold 3).

For the "lower" unique amino acid threshold (equal to 2), the best Z-score threshold for each method is as follows: CAPS, 1; PlotCorr, 2; MI, 1; Waddell – MI, 0; Waddell – Kappa, 3 (see Table 6.3). For the "higher" unique amino acid threshold (equal to 4), the best Z-score threshold for each method is as follows: CAPS, 1; PlotCorr, 2; MI, 1; Waddell – MI, 3; Waddell – Kappa, 3. The only method for which there is a difference in the best Z-score threshold between the two amino acid thresholds is for Waddell – MI (see Table 6.4).

At this point, we have fixed the Z-score type (original values), the unique amino acid thresholds ("lower" = 2, "higher" = 4), and the Z-score thresholds (for each unique amino acid threshold and co-evolution detection method). These values are set for the rest of this chapter. Now we can break down the results in terms of other parameters, such as the number of sequences in the alignment, alignment length, rate variation parameter, and percentage of sites that are co-evolving, and see what effect the values of these parameters have on results.

6.6 Results by number of sequences

We first break down the results according to the number of sequences in the input alignment. There were three values of this parameter in the simulations:

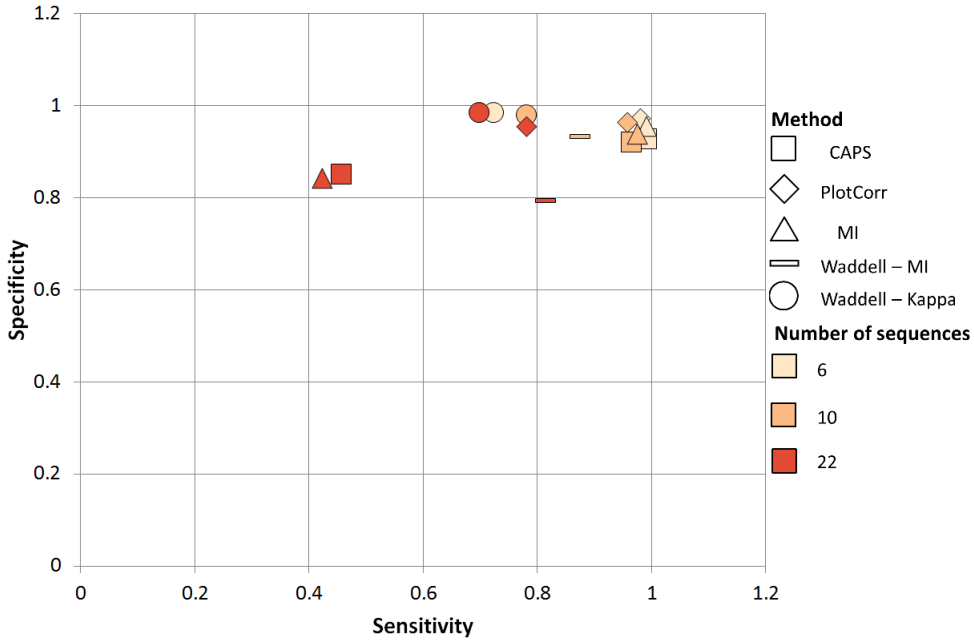

Figure 6.8: Mean sensitivities and specificities for unique amino acid threshold 2 (with Z-score type fixed to original values Z-score, and each method's Z-score threshold fixed as detailed in the text). Results are broken down by number of sequences in the alignment.

6, 10, and 22. We shall now analyse the results for each unique amino acid threshold. All of the p-values for all tests performed are in Appendix A.6.

6.6.1 "Lower" threshold

For the "lower" unique amino acid threshold (= 2), we see the average results shown in Figure 6.8. Averages are shown simply to aid understanding; statistical tests, the results of which are discussed below, are run on all data points.

To ensure the reliability of these results, statistical tests were performed comparing the performance of each pair of co-evolution detection methods. After confirming that the data sets were all non-Normal, both the two-sided Wilcoxon signed-rank test and the two-sided Sign test were performed (the latter in case the data was not symmetrically distributed). The p-values from both tests were examined for each comparison, and in the case where the two tests disagreed, the actual data distributions were examined to determine which test's result was preferred. Once the differences were confirmed, one-tailed tests were performed to confirm exactly which method had the better performance in each pair. This method was used throughout this chapter for the statistical tests. (When comparing groups where these groups were not the same size, we would use the unpaired version of this test, the Wilcoxon

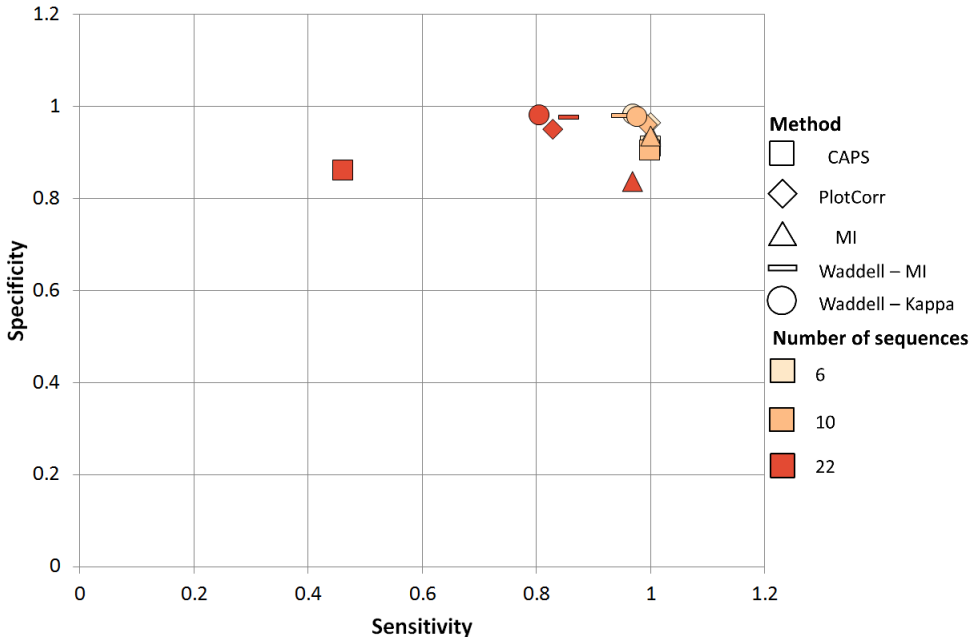

Figure 6.9: Results for unique amino acid threshold 4 (with Z-score type fixed to original values Z-score, and each method's Z-score threshold fixed as detailed in the text). Results are broken down by number of sequences in the alignment.

rank-sum test.) From this statistical testing, it was found that overall, each method's performance relative to the others is constant across different numbers of sequences, except in the cases of comparing CAPS and Waddell – MI, and of comparing MI and Waddell – MI. It was found that overall, regardless of the number of sequences in the alignment, Waddell – Kappa and PlotCorr are the best-performing methods. Which methods perform next-best depends on the number of sequences.

Examining the problem the "opposite way around", i.e. examining which numbers of sequences each method performs best for, CAPS, PlotCorr, and MI all perform best for a smaller number of sequences. In contrast, the parametric methods perform best for different numbers of sequences depending on which statistic (sensitivity, specificity, or precision) you wish to maximise. Preferring larger specificities and precisions overall, for Waddell – Kappa specifically, having 22 sequences gives the best performance overall, and for Waddell – MI, having 6 sequences gives the best performance, maximising both statistics as much as possible.

6.6.2 "Higher" threshold

For the "higher" unique amino acid threshold (= 4), we see the average results shown in Figure 6.9. Averages are shown simply to aid understanding;

statistical tests, the results of which are discussed below, are run on all data points.

Again, statistical tests were performed, comparing the performance of each pair of co-evolution detection methods using all results. From this statistical testing, it was found that overall, each method's performance relative to the others is constant across different numbers of sequences, except in the cases of comparing CAPS and MI, and of comparing Waddell – Kappa and Waddell – MI. It was found that the parametric methods are the best overall; which one exactly is better depends on the number of sequences in the alignment. PlotCorr is the next best-performing method, and then CAPS is the worst (for 6 sequences it performs equally to MI).

Examining the problem the "opposite way around", i.e. examining which numbers of sequences each method performs best for, CAPS, PlotCorr, and MI perform best for smaller numbers of sequences. However, overall, both of the parametric methods individually perform with around the same performance for all 3 numbers of sequences.

Overall, Waddell – Kappa is the best-performing method across both unique amino acid thresholds.

6.7 Results by alignment length

Now we break down the results according to the alignment length (number of columns). The values of this parameter used in the simulations were 50, 600, and 1200. We shall now look at the effects of varying this parameter on the results obtained for the two unique amino acid thresholds. All of the p-values for all tests performed are in Appendix A.7.

6.7.1 "Lower" threshold

For the "lower" unique amino acid threshold (= 2), we see the average results shown in Figure 6.10.

From statistical testing using all data points, it was found that overall, each method's performance relative to the others is constant across different alignment lengths. Using the results of the pairwise statistical comparisons, an ordering can be made of the methods in terms of overall performance: Waddell – Kappa, PlotCorr, MI, Waddell – MI, CAPS.

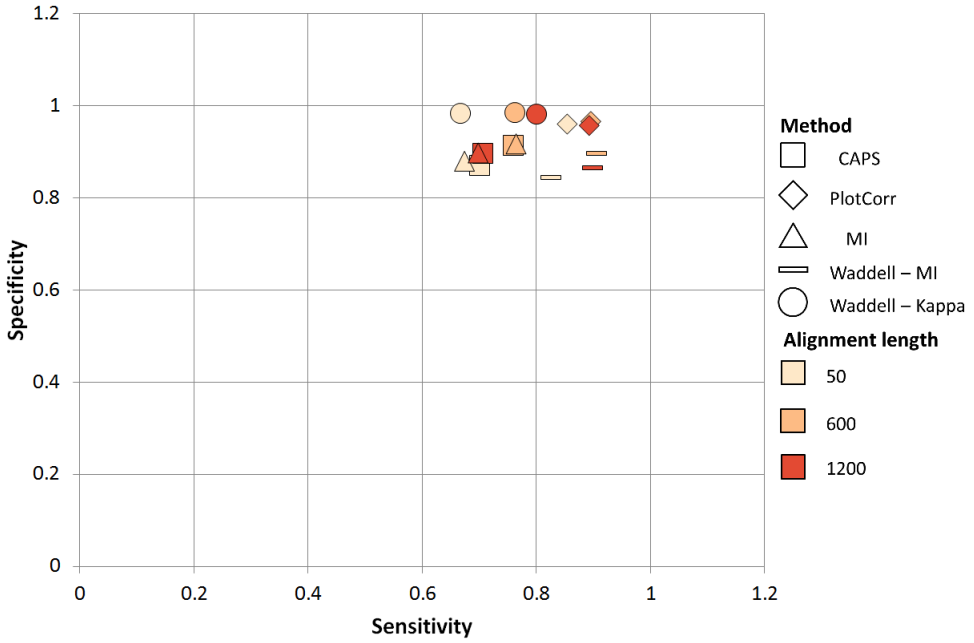

Figure 6.10: Results for unique amino acid threshold 2 (with Z-score type fixed to original values Z-score, and each method's Z-score threshold fixed as detailed in the text). Results are broken down by alignment length.

Using the same kind of statistical testing to work out which alignment lengths each method performs best for, it is difficult to find patterns that describe which alignment lengths each method performs best for; there appear to be differences between the performances for different lengths for certain methods, but these differ depending on the test statistic (sensitivity, specificity, precision) that is being considered. These effects could be due to chance, but it is interesting to note that the Z-score for a pair of positions depends not only on the correlation calculated for that pair (whether that be CAPS, MI, etc.) but also on the correlations calculated for all the other pairs of positions in the alignment being considered, so maybe having more or fewer correlations accounted for in the mean and standard deviation for a Z-score calculation will have an effect. For all of the methods, we are able to obtain higher precision values for smaller alignment lengths. For Waddell – Kappa, higher sensitivities can be obtained with a longer alignment.

For Waddell – Kappa, it can be seen that as the alignment length increases, the number of true positives increases, and the number of false negatives decreases – this could explain the increase in sensitivity. For all methods, the ratio of true positives to false positives decreases as the alignment length is increased, which explains the decrease in precision.

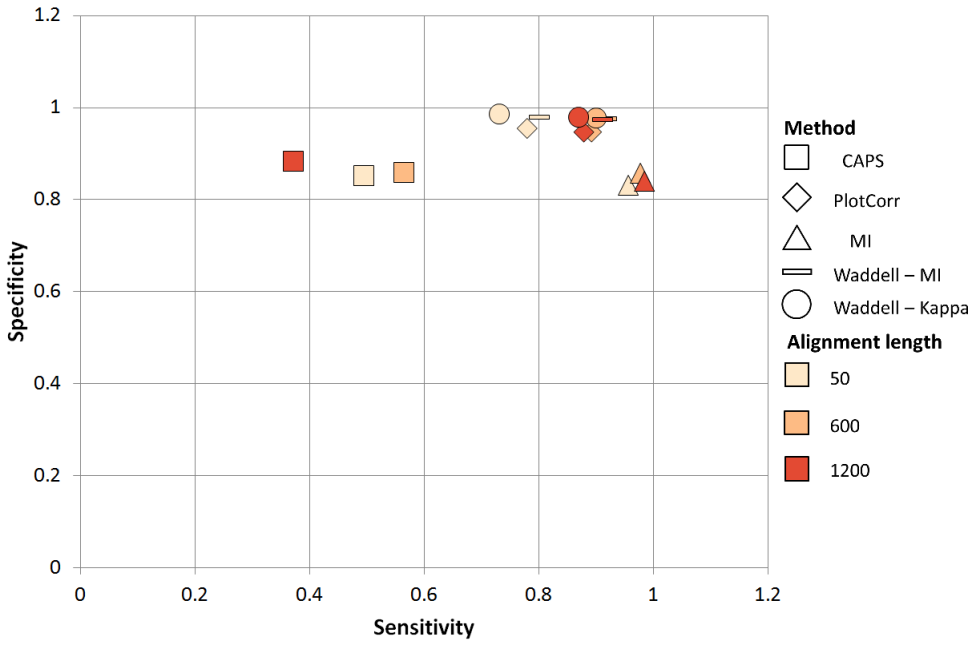

Figure 6.11: Results for unique amino acid threshold 4 (with Z-score type fixed to original values Z-score, and each method's Z-score threshold fixed as detailed in the text). Results are broken down by alignment length.

6.7.2 "Higher" threshold

For the "higher" unique amino acid threshold (= 4), we see the results shown in Figure 6.11.

As with the "lower" unique amino acid threshold, the relationships between pairs of methods in terms of their relative performance is constant regardless of the alignment length being considered. An ordering can be made, once again, to show the performance between all of the methods: Waddell – Kappa, Waddell – MI, PlotCorr, MI, CAPS.

Again, observing the performances for each method individually on different alignment lengths, we see that no patterns can really be derived. For all methods, the ratio of true positives to false positives decreases as the alignment length is increased, which explains a decrease in precision. There are larger specificities for a decrease in alignment length for Waddell – MI only; the ratio of true negatives to false positives decreases as the alignment length increases.

Looking at the relative performance of the methods, comparing results between the "lower" and "higher" UAA thresholds, the only real difference we see is that Waddell – MI's performance is increased for a higher UAA threshold. Overall, Waddell – Kappa performs the best over both UAA thresholds.

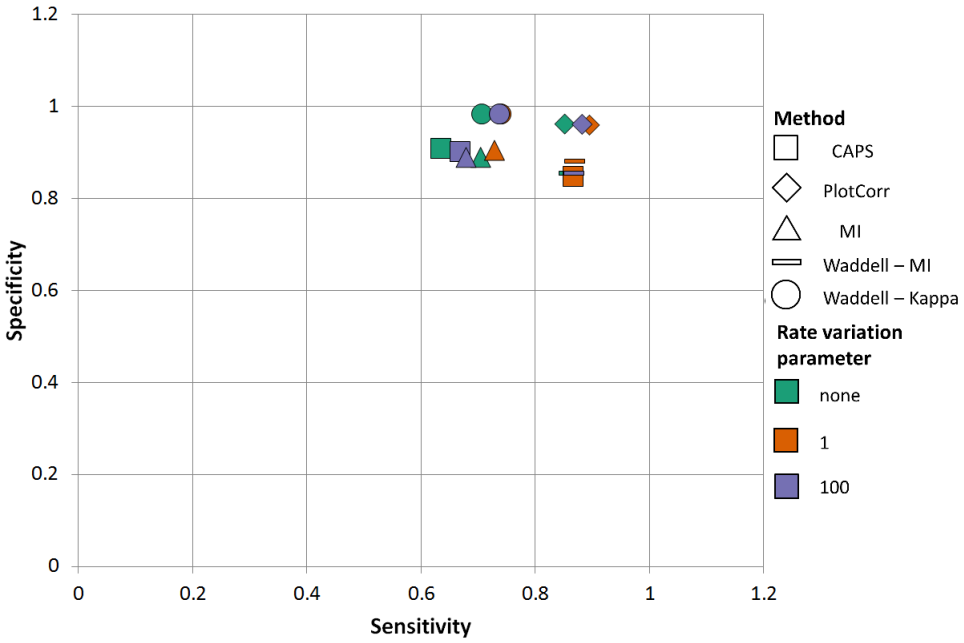

Figure 6.12: Results for unique amino acid threshold 2 (with Z-score type fixed to original values Z-score, and each method's Z-score threshold fixed as detailed in the text). Results are broken down by rate variation parameter.

6.8 Results by rate variation parameter

Now we break down the results according to the value of the rate variation parameter, α. The α parameter to the Gamma distribution influences the amount of variation between the rates of mutation of the different columns (the higher the rate of mutation, the more likely that a column will be chosen to mutate by the simulation program). We used two values of α: 1, which means the distribution is L-shaped, so many sites have low rates of mutation and a smaller number have higher rates, and 100, which means the Gamma distribution looks more Normal, so most sites have an intermediate rate of mutation, while a smaller number have much higher or lower values. We also ran simulations with no Gamma distribution (called "none" on the graphs), which means that all columns have the same rate of mutation. We shall now look at the effects of varying this parameter on the results obtained for the two unique amino acid thresholds. All of the p-values for all tests performed are in Appendix A.8.

6.8.1 "Lower" threshold

For the "lower" unique amino acid threshold (= 2), we see the average results shown in Figure 6.12. It should be noted straight away that all of the points on the graph have sensitivity of at least 0.6, and specificity of at least 0.8, so there does not appear to be much spread over all rate variations and methods.

Performing statistical tests between all methods for each rate variation parameter value (no rate variation, 1, 100), each method's performance relative to every other method is constant, regardless of whether there is any rate variation, and the shape of the rate variation if so. This means that none of the methods are impacted by the type of rate variation. From these comparisons, we can, once again, come up with an ordering of the methods according to their overall performance in comparison to all the other methods. This ordering is: Waddell – Kappa, PlotCorr, MI, Waddell – MI, CAPS. Although there is some difference in each method's performance for different statistics (sensitivity, specificity, precision) depending on the rate variation parameter, this parameter does not make a difference to the overall performance.

We then examine the problem the "opposite way around", i.e. attempt to work out which rate variation parameter values each method performs best for. Doing so it can often be found that for a co-evolution detection method, the pair of rate variation values being compared (none, 1, 100) only differ on one statistic (sensitivity, specificity, precision). In this case, the two rate variation values are said to produce approximately the same results – this makes explaining the results easier. PlotCorr and Waddell – Kappa have the same performance regardless of the amount of mutation rate variation among the sites. MI and Waddell – MI perform best for the rate variation parameter 1 (most sites having a low substitution rate, and a smaller number of sites having a higher rate), then no rate variation, then for parameter 100 (a Normal-like distribution) they have the worst performance. This makes a lot of sense as Mutual Information gives columns with a higher number of mutations a higher score – it "rewards" sites with high entropy. If most sites have a low substitution rate, then it makes sense that it would be easiest to pick out the sites with the higher rate, as these are the ones that are more likely to be co-evolving. On the other hand, if most sites have similar mutation rates, then it would make the task of detecting co-evolving sites more difficult. CAPS performs best for rate variation parameter 1, like for MI and Waddell – MI, but it performs approximately the same for no rate variation as for rate variation parameter 100.

6.8.2 "Higher" threshold

For the "higher" unique amino acid threshold (= 4), we see the average results shown in Figure 6.13. Overall, there is much more of a spread in the performance of the different methods over different values of the Gamma parameter, as seen on the graph.

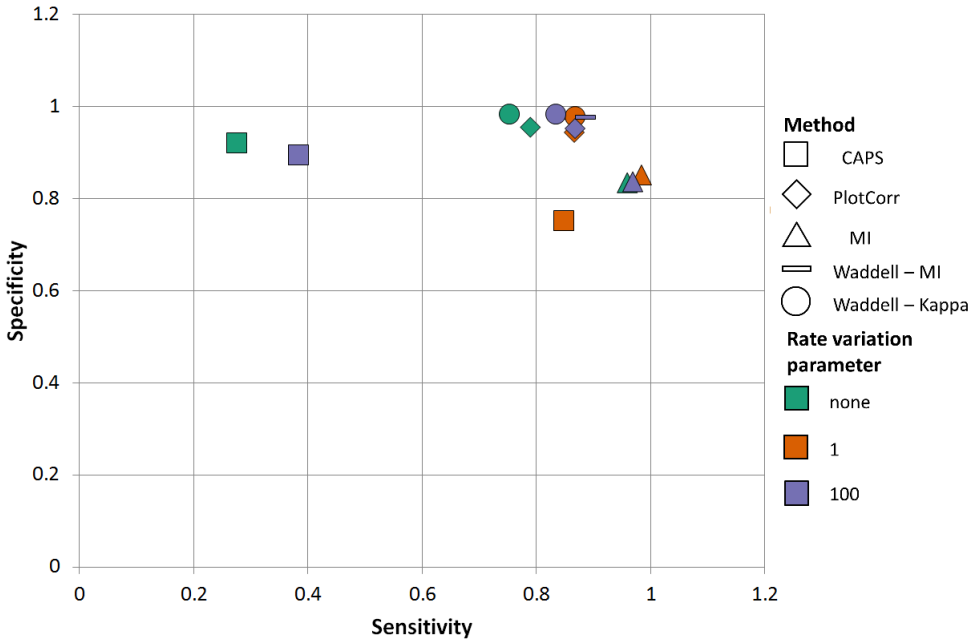

Figure 6.13: Results for unique amino acid threshold 4 (with Z-score type fixed to original values Z-score, and each method's Z-score threshold fixed as detailed in the text). Results are broken down by rate variation parameter.

Again performing statistical comparisons between all pairs of methods, we can derive an ordering to the methods according to their overall performance in comparison to the others. The ordering found was: Waddell – Kappa, Waddell – MI, PlotCorr, MI, CAPS. This time, Waddell – MI performs better with a higher number of unique amino acids.

Once again examining the problem in terms of working out which rate variation parameter values each method performs best for, PlotCorr and Waddell – Kappa, as with the "lower" unique amino acid threshold, have approximately the same performance for each of the rate variation parameter values. For CAPS, the order of performance is now parameter value 1, then 100, then no rate variation. This means that having a rate variation parameter of 100 now gives better performance than having no rate variation at all. For MI, having rate variation parameter 1 still performs the best, but having no rate variation now gives similar performance to having rate variation parameter 100. For Waddell – MI, all three values of the rate variation parameter now give approximately the same performance. This is likely due to the fact that having a higher threshold UAA value means that more sites will have a higher amino acid diversity anyway, giving better performance to a method like MI which gives higher scores to positions with high entropy (equal to a higher number of unique amino acids).

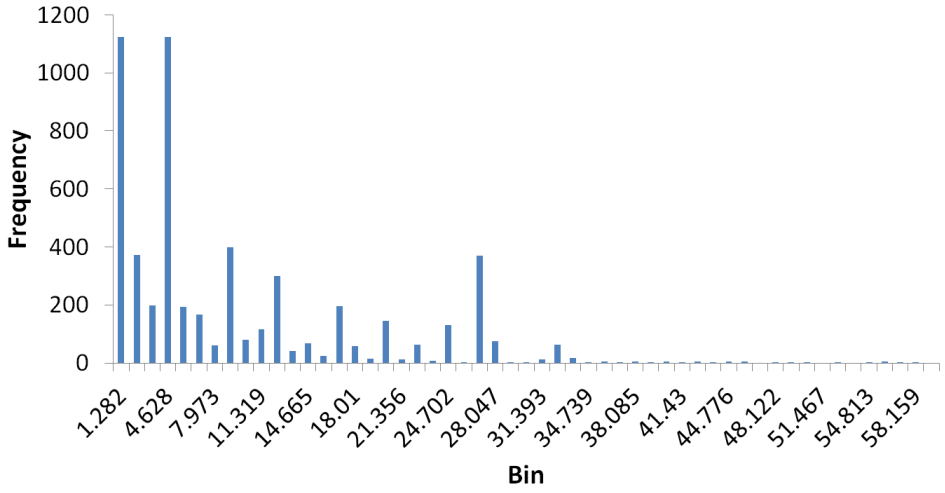

Figure 6.14: Histogram showing the percentages of sites co-evolving for simulated data with unique amino acid threshold 2.

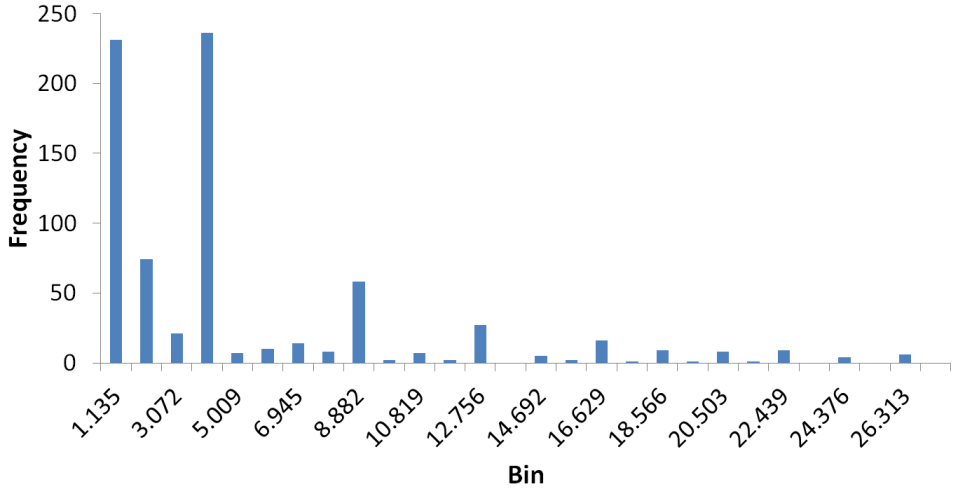

Figure 6.15: Histogram showing the percentages of sites co-evolving for simulated data with unique amino acid threshold 4.

Overall, we find that PlotCorr and Waddell – Kappa are less influenced by the amount of rate variation, which is good because we do not actually know how much rate variation a protein may have in real data.

6.9 Results by percentage of sites co-evolving

Finally, we look at the results from the perspective of the percentage of sites in the alignment that are co-evolving, calculated as (columns co-evolving in the alignment/total number of columns in the alignment) * 100. To choose the bins that the percentages were placed into for the sake of the comparison, we inspected the distributions of percentages in the simulated alignments, for UAA thresholds 2 (Figure 6.14) and 4 (Figure 6.15) respectively. From doing so it was apparent that bins of width 5% were appropriate, and after a

certain number (35% for UAA 2, and 15% for UAA 4) the rest of the percentages were put in a final bin, when it became known that only a few cases would satisfy this largest bin. All of the p-values for all tests performed are in Appendix A.9.

6.9.1 "Lower" threshold

For the "lower" unique amino acid threshold (= 2), it is difficult to plot all results on one graph, so we just go straight to the results of statistical comparisons.

Again, statistical testing was used to work out which methods perform the best for each parameter value (percentage bin). The bins that the results were placed in are: 0–5%, 5–10%, 10–15%, 15–20%, 20–25%, 25–30%, 30–35%, $>$ 35%. Each pair of co-evolution detection methods was compared in terms of their results to find the best-performing in each pair. From these comparisons it is possible to create an ordering of the methods for each percentage bin. It should be noted that for most percentage bins, the ordering is exactly the same, i.e. most of the time, the methods which perform the best for one percentage bin, perform the best for most others. The only methods whose performance relative to the other methods was different based on which percentage bin is being considered are MI and Waddell – MI. For the percentage bins 0–5%, 5–10% and 10–15%, the relative ordering of the methods based on performance (from best-performing overall to worst) is: Waddell – Kappa, PlotCorr, MI, Waddell – MI, CAPS. For 15–20%, 20–25%, 30–35%, and $>$ 35%, Waddell – MI and MI swap positions (Waddell – MI performs better than MI). For 25–30%, MI and Waddell – MI perform approximately the same.

Examining the problem the "opposite way around" i.e. finding which percentage bins each method performs best for, it is apparent that this depends on the statistic being examined (sensitivity, specificity, or precision). It is, however, interesting to note where certain methods perform best for smaller percentage bins. This occurs in the cases of all sensitivities except for MI, for PlotCorr precisions, and for Waddell – MI specificities.

6.9.2 "Higher" threshold

For the "higher" unique amino acid threshold (= 4), it is difficult to plot all results on one graph, so we just go straight to the results of statistical comparisons.

Again, comparing the performance of each pair of co-evolution detection methods to determine an overall order to the methods, we find that the relative performance of the methods is the same almost throughout. For 0–5%, 5–10%, and $> 15\%$ bins, the order is given by: Waddell – Kappa, Waddell – MI, PlotCorr, MI, CAPS. For the 10–15% bin, Waddell – MI performs better than Waddell – Kappa. The rest of the overall performances are the same.

Examining the problem the "opposite way around", i.e. finding the percentage bins that the methods perform best for, there appear to be more obvious rules than for UAA threshold 2. In particular, for all methods, larger percentage bins give better performance for precisions and specificities, and smaller percentage bins give better performance for sensitivities. Having better performance for larger bins for precisions makes sense as this implies a larger number of true positives are being detected. Also, having better performance for smaller bins for sensitivities makes sense in that this implies fewer false negatives in the equation, but the specificities pattern (improved performance for larger bins) is not as we would expect. Generally, all of the methods are affected by the percentage of co-evolving sites although not in the way we would expect.

The differences between UAA 2 and UAA 4 in terms of method performance is that Waddell – MI performs better for the "higher" UAA threshold; this is likely due to the method's use of Mutual Information, which gives higher MI scores to sites with a higher diversity of amino acids. Also, for both UAA thresholds, all methods are somewhat affected by the percentage of co-evolving sites present. This is as we would expect. The parametric methods and PlotCorr perform the best across both unique amino acid thresholds.

6.10 Combining methods

In the previous sections we found that the best-performing methods were PlotCorr, Waddell – MI, and Waddell – Kappa. We shall now investigate what happens when we combine these methods in different ways in an attempt to improve the overall performance. We shall use the results we have already generated for our simulated data, but taking a different approach towards counting the number of true positives, true negatives, etc. We refer the reader to Appendix A.10 for an example illustrating how the process of method combination works.

Table 6.5 shows the average sensitivity, specificity, and precision for different combinations of the methods. Comparing each possible combination

Combination	Low threshold	High threshold
PlotCorr, Waddell – MI, Waddell – Kappa		
Sensitivity	0.697848	0.715937
Specificity	0.988185	0.990221
Precision	0.185273	0.100020
PlotCorr, Waddell – MI		
Sensitivity	0.800805	0.749168
Specificity	0.970537	0.988413
Precision	0.120532	0.089835
PlotCorr, Waddell – Kappa		
Sensitivity	0.697848	0.718135
Specificity	0.988185	0.988703
Precision	0.185273	0.088870

Table 6.5: Overall results for the combinations of different methods.

of the three methods against the five individual methods compared within this chapter (CAPS, PlotCorr, MI, Waddell – Kappa, Waddell – MI), for both unique amino acid thresholds it is always the case that the combination is better than the individual methods in terms of precision, while the individual methods are better than the combinations for specificities. For sensitivities, each individual method is better than the combinations, except for Waddell – Kappa in the case of the "lower" unique amino acid threshold, and CAPS in the case of the "higher" unique amino acid threshold. We conclude that it is ample to use the individual methods themselves.

6.11 Summary

Overall, we have found that the original values Z-score type is the preferred approach; however you look at the results it excels over the shuffle Z-score type. This is very interesting as the original values Z-score requires fewer calculations. However it does use information from the entirety of the alignment, while the shuffle Z-score does not; the original values Z-score approach compares the result at one pair of positions to the average at all pairs of positions.

In this chapter, we considered the effect of varying different simulation parameters (number of sequences per alignment, alignment length, rate variation parameter, percentage of sites co-evolving). Whichever parameter was being varied, we found that overall, PlotCorr and the two parametric methods performed the best.

The methods we have chosen to implement are PlotCorr and the two parametric methods. We did not want to pick between the two parametric methods

and decided to just implement both because there is enough in common between the two methods such that it would require little effort to build both statistics into a single parallel implementation.

We have found that it is ample to consider the co-evolution detection methods individually, as opposed to combining them, so we continue to consider them individually from now on.

In this chapter we compared all of our methods, and concluded that PlotCorr and the two parametric methods (Waddell – MI and Waddell – Kappa) are worth pursuing. We shall implement these methods in parallel in the next chapter.

Chapter 7

Parallelisation of Methods using CUDA

In the previous chapter the methods with the best performance in terms of their ability to detect molecular co-evolution were found; these are PlotCorr and the two parametric (Waddell – MI and Waddell – Kappa) methods. In this chapter we shall detail the parallelisation of these methods. The step this chapter covers is shown in Figure 7.1. We begin by describing the APOD (Assess, Parallelise, Optimise, Deploy) cycle, which was used to inform the parallelisation of the code. We shall then describe the experimental set-up used to measure the execution times of our implementations, and compare the running times of these (the implementations being achieved in serial on the CPU, and in parallel on two CUDA-enabled graphics cards: a GTX 660 Ti and a Titan X). Finally, we give a cost/benefit analysis of the two graphics cards alongside a CPU-only system and a supercomputer in terms of an "average", and "large", co-evolution detection task.

7.1 Approach taken to parallelisation

In Section 2.9, we introduced the CUDA architecture and the reasons for wanting to develop new parallel implementations of methods for detecting molecular co-evolution. Briefly, until this point it has been impractical to analyse the entire human proteome to detect co-evolving sites due to the prohibitive amount of time this would take – this is the motivation for increasing the performance of methods for co-evolution detection. In this section we shall detail the APOD approach used toward parallelising the chosen methods. This general methodology can be used when improving any parallel code, although the details given later are particular to the exact set-up used (developing on a Windows PC with Visual Studio, using CUDA to implement each method).

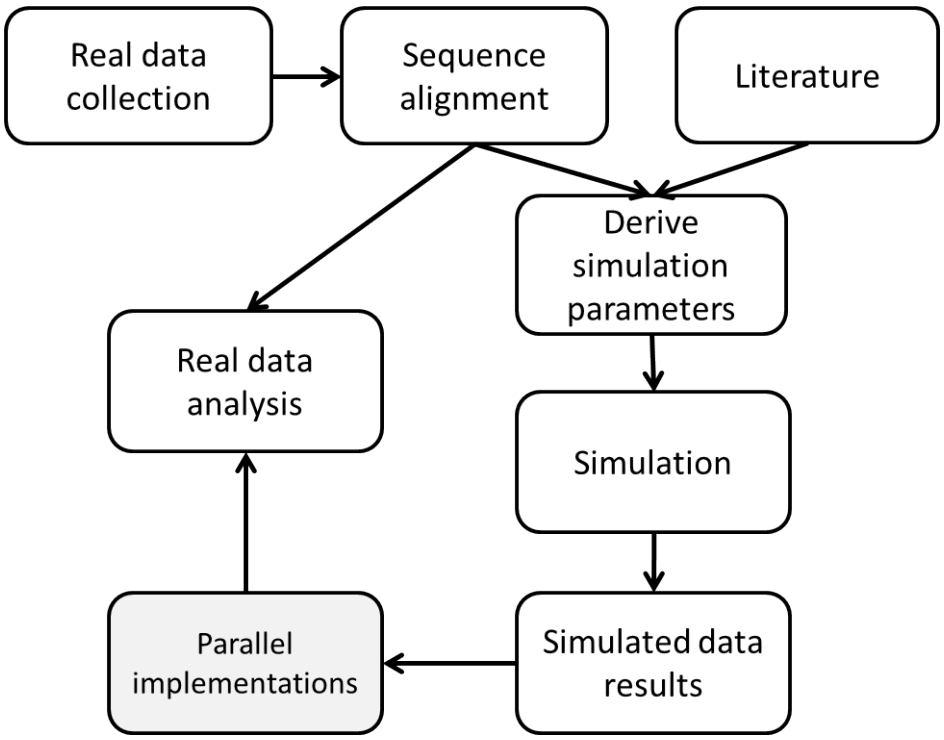

Figure 7.1: This chapter covers the parallelisation of the co-evolution detection methods chosen in the previous chapter (shaded box).

APOD stands for Assess, Parallelise, Optimise, Deploy [14], and although it originated at NVIDIA (the company which created the CUDA programming language and architecture), it can feasibly be applied to any architecture where the parallelism arises through the use of a parallel programming language. We shall now describe the activities performed at each stage.

Assess The code is studied (often in an automated manner, such as by using a profiling tool), and the code's bottlenecks are identified. According to Bradley [14], we need to take into account which type of scaling we wish to focus on, strong or weak. Strong scaling 'is a measure of how, for a given problem size, performance changes as more processors are added to the system' [14, para. 6]. In contrast, weak scaling 'is a measure of how the performance *per unit of work* changes as more processors are added' [14, para. 6]. Strong scaling therefore means fixing the problem size, and improving on the performance of solving the problem given that fixed problem size, and weak scaling is about increasing the problem size as increased processing power becomes available [119]. For this thesis, the appropriate type of scaling is strong scaling – it has always (in recent times) been possible (at least in theory) to calculate the likely co-evolving sites within the human proteome, but this would have taken a prohibitively long time. CUDA is a potentially cost-effective means of achieving the necessary performance increase that could allow us to solve this problem in a reasonable amount of time, and it is the

technology that we use here.

Parallelise Having identified the bottlenecks in the previous stage, the next step is to parallelise the parts of the code where the bottlenecks occur.

Optimise This stage involves repeating parts of the Assess stage; the goal now is to ensure the parallel code implemented in the previous stage is as fast as possible. We can profile our parallel code after applying various optimisations, such as changing the type of memory used (global versus shared memory), and use the results of profiling to improve.

Deploy Moving updates to code from development to production-ready applications is something not relevant to this project, but would be relevant to a commercial project.

Normally the cycle begins when the code is still serial, so the most time-consuming parts are identified and then improved through parallelisation, before moving onto parallelising other parts of the code as necessary. However, we already had a good idea of where improvements to the code could be made, so we began from the outset with implementing each method in CUDA as much as possible. We did this because of the general rule that moving data between the PC and the GPU is time-consuming, so running code on the GPU (in parallel) can result in better overall performance if doing so means avoiding data transfer, even if that computation would have better performance in a serial implementation.

Now that we have described the APOD cycle generally, we shall detail how the CUDA implementations were carried out, with reference to the cycle.

As mentioned previously, we started by parallelising the code as much as possible. This meant beginning at the Assess stage with basic parallel implementations of the methods. In order to improve on the first basic versions, we used Visual Studio's NSIGHT profiler, which shows how much time is being spent on each method, allowing one to focus on the bottlenecks in the code. After identifying the bottlenecks from the timeline provided by NSIGHT, we picked the function(s) which were taking the longest, and improved on these, before timing the code again and repeating the cycle.

We shall now introduce, for each of the methods implemented, the various functions each method uses, and then list the ways in which we took advantage of CUDA's architecture to improve the performance of each method.

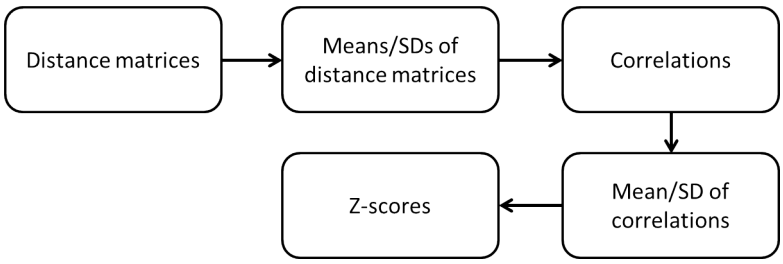

Figure 7.2: The functions involved in the PlotCorr method.

7.1.1 PlotCorr

First, for each function within the PlotCorr algorithm, we shall briefly describe what this function does (Figure 7.2 shows how these functions fit together to make the overall PlotCorr method). Then, more detail will be given on how the APOD cycle was used to improve the performance of the PlotCorr method overall.

Method descriptions

First we calculate the **distance matrix** for each column of the alignment. The values within each distance matrix are taken from a substitution matrix such as PAM or BLOSUM, so, for example, if we were to calculate the distance matrix for the third column in the alignment, then the value in the first column, in the second row of this distance matrix would be found by looking at the third column of the alignment, and picking out the amino acids at the first and second rows of the alignment. We then look up the value for the similarity between these amino acids in a substitution matrix such as PAM or BLOSUM, described in Chapter 2, and this is the value used in the distance matrix.

For each distance matrix just calculated, calculate the **mean and standard deviation (SD) of the values within each matrix**.

Calculate the **correlation** for each pair of positions in the alignment.

Calculate the **mean and standard deviation of the correlations** calculated in the previous step.

Calculate the **Z-score** for each pair of positions in the alignment.

Each of the statements in bold corresponds to one or more functions in the code, and can be thought of as one coherent piece of the puzzle.

Application of APOD

In the previous section, the various steps which make up the PlotCorr method were briefly described. In this section, more detail will be given on how the APOD cycle was used to parallelise PlotCorr. Each "version" that the code went through shall be discussed.

Version 1 The initial implementation of PlotCorr sought to parallelise as much code as possible in a relatively naive fashion. This would then be improved upon in subsequent versions.

The version of the distance matrix calculation code used in this initial iteration was retained for all further versions of PlotCorr. A thread is allocated to each column in the alignment, so each thread inspects the characters within its column and constructs a distance matrix.

When it comes to calculating the mean and standard deviation of a set of values, there are several approaches. In this case, we wish to calculate the mean and standard deviation of the values within each distance matrix. For this first version of the code, the two-pass method was used. It is called this because the mean is calculated in one pass-through of the data, and then the standard deviation is calculated in a second pass-through of the data. The equation used to calculate the mean should be easily recognisable:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{7.1}$$

where $\overline{x}$ is the mean of all values, and x_i is the ith item in the list of n items. Knowing the mean, the standard deviation can be calculated using:

$$\sigma(x) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}} \tag{7.2}$$

A separate function was used to calculate the means and the standard deviations. For each of these functions, only one thread is used (these CUDA functions are essentially serial).

Moving onto calculating the correlations themselves (the approach to doing so being particular to PlotCorr), each thread calculates the correlation for a different pair of positions within the alignment.

The mean and standard deviation of all of the correlation values is completed in one method (this only needs to be done once for all of the correlation values, as opposed to the distance matrix mean/SD calculations, which

	Name	Launches	Device %	Total (µs)
1	calcMeanSD	1	4.04	121,579.552
2	calcStandardDeviations	1	0.98	29,669.035
3	calcMeans	1	0.28	8,568.061
4	calcCorrelations	1	0.03	789.906
5	calcDistMatrixParallelPerColumn	1	0.00	43.944
6	origZScorePerElementAll	1	0.00	24.069

Figure 7.3: NSIGHT profiler information for Version 1 of the PlotCorr code.

are completed for each distance matrix individually), hence why a different approach was taken. For this version of the code, the naive one-pass method was used. This method only requires one pass-through of the data to calculate both the mean and standard deviation. At each step, we take the current integer, add it to a sum, and then square it and add it to another sum. The sum of the values is divided by the number of values to obtain the mean, and the sum of the squared values is divided by the number of values, and then the square root is taken of this sum, to obtain the standard deviation.

Finally, the Z-score is calculated for each pair of columns; the implementation of the function for completing this calculation is the same as that used in every version of the PlotCorr code. Each thread calculates a Z-score using the equation:

$$Z_{ij} = \frac{(x_{ij} - \overline{x})}{\sigma} \tag{7.3}$$

We subtract the mean of the correlations from each individual correlation value, and then divide by the standard deviation of the correlations. Having already calculated the mean and standard deviation, we only need to do the subtraction and division, which can be easily done in parallel on an element-by-element basis.

Profiling Version 1 in NSIGHT within Visual Studio, we found that most time was spent calculating the mean and standard deviation of the correlations (see Figure 7.3). Before improving upon this function specifically, however, we changed the functions that perform calculations on pairs of columns so that they only worked on those pairs i, j where $i < j$, which eliminates repeated calculations.

	Name	Launches	Device %	Total (µs)
1	calcMeanSDNoRepeats	1	4.06	123,930.553
2	calcStandardDeviations	1	1.28	39,015.740
3	calcMeans	1	0.37	11,200.140
4	calcCorrelationsNoRepeats	1	0.02	575.246
5	calcDistMatrixParallelPerColumn	1	0.00	57.867
6	origZScorePerElementAll	1	0.00	17.635

Figure 7.4: NSIGHT profiler information for Version 2 of the PlotCorr code.

Version 2 In this version, the functions that performed calculations on pairs of columns (the calculation of correlations, calculating the mean and standard deviation of correlations) were changed so that they only performed their calculations on (length * (length - 1))/2 position pairs (where length is the number of columns in the alignment). This is equal to only calculating the correlation value for pairs i, j where $i < j$. Upon profiling the resulting code in NSIGHT, the time was still mostly being spent on calculating the mean and standard deviation of the correlations (see Figure 7.4), and so this was focused on in the next code iteration.

Version 3 The function to calculate the mean and standard deviation of the correlations was replaced with a two-pass approach:

1. Calculate the mean of the correlations, $\overline{x}$, by summing the values using reduction (a parallel algorithm, explained below), and then dividing the result by the number of correlations

2. Calculate, for each correlation x, the value of $(x - \overline{x})^2$. This can easily be done in parallel

3. Calculate the mean of the squared differences just calculated, using the same reduction algorithm used in Step 1

4. Take the square root of the mean from Step 3 to find the standard deviation of all the correlation values

The algorithm for reduction, used in steps 1 and 3 above, takes a set of elements and a binary associative operator as input, and returns the result of performing that operator on those inputs [197]. "Binary" means the operator performs the operation on two elements. "Associative" means the result of performing the operator on the two inputs is the same, regardless of the order in which those elements are presented. For example, given an input array:

```
[27, 29, 66, 11, 8, 52, 71, 53, 26, 69]
```

The result of performing the binary associative operator addition on this array is 412, which is just the result of adding all the elements together. What is crucial is this can be done in parallel.

```
1  __global__ void reduction_explicitloop(float *d_out
       , const float *d_in, size_t N) {
2
3          extern __shared__ float sPartialsReduction
       [];
4          float sum = 0.0f;
5          const int tid = threadIdx.x;
6          for (size_t i = (blockIdx.x * blockDim.x) +
        tid; i < N; i += blockDim.x * gridDim.x) {
7                  sum += d_in[i];
8          }
9          sPartialsReduction[tid] = sum;
10         __syncthreads();
11
12         for (int activeThreads = blockDim.x >> 1;
       activeThreads; activeThreads >>= 1) {
13                 if (tid < activeThreads) {
14                         sPartialsReduction[tid] +=
       sPartialsReduction[tid + activeThreads];
15                 }
16                 __syncthreads();
17         }
18
19         if (tid == 0) {
20                 d_out[blockIdx.x] =
       sPartialsReduction[0];
21         }
22
23 }
24
25
26 void reduction(float *d_out, float *d_intermediate,
        const float *d_in, size_t N, int numBlocks, int
        numThreads) {
27
```

```
28          unsigned int sharedSize = numThreads *
    sizeof(int);
29          reduction_explicitloop<<<numBlocks,
    numThreads, sharedSize>>>(d_intermediate, d_in, N
    );
30          gpuErrchk(cudaPeekAtLastError());
31          gpuErrchk(cudaDeviceSynchronize());
32          reduction_explicitloop<<<1,numThreads,
    sharedSize>>>(d_out, d_intermediate, numBlocks);
33          gpuErrchk(cudaPeekAtLastError());
34          gpuErrchk(cudaDeviceSynchronize());
35
36 }
```

Listing 7.1: Reduction code

Listing 7.1 shows the code for the two functions relevant to this step. This code was released with 'The CUDA handbook' [197] under the 2-clause BSD licence – we have made some changes to this code, mostly reformatting and variable renaming.

The `reduction` function is called first, where we work out the amount of shared memory we need. We then call the `reduction_explicitloop` function, which creates a shared memory array, `sPartialsReduction`, of the same size as the shared memory allocation which was calculated in the `reduction` function. The for loop in line 6 moves over the data (the `d_in` array), in chunks of size `blockDim.x * gridDim.x`, which allows for values of `N` larger than the number of threads we have running. Once we have performed the sum in the for loop, we store this sum at the position represented by our thread index in `sPartialsReduction`. The next for loop in this function (line 12) starts with the `activeThreads` variable, but with its value bit-shifted once to the right (equal to dividing it by 2). Within the for loop itself, we check whether the thread ID is less than `activeThreads`; if so, we add to the value at the position represented by our thread index in `sPartialsReduction`, the value within the same array, `activeThreads` values away. This means that in each iteration of the loop, we add a value closer and closer to our own in terms of index. When we complete the loop, the thread with index 0 sets the output value for our block to the final result.

In the `reduction` function, we perform `reduction_explicitloop` once, check for errors and synchronise, and perform the reduction once more

	Name	Launches	Device %	Total (µs)
1	calcStandardDeviations	1	1.10	29,625.116
2	calcMeans	1	0.32	8,559.576
3	calcCorrelationsNoRepeats	1	0.02	432.147
4	reduction_explicitloop	4	0.00	56.459
5	calcDistMatrixParallelPerColumn	1	0.00	43.944
6	origZScorePerElementAll	1	0.00	13.634
7	valueMinusMeanParallel	1	0.00	11.394

Figure 7.5: NSIGHT profiler information for Version 3 of the PlotCorr code.

on our block of intermediate values obtained from the first run (and check for errors and synchronise). After this function is performed, we then divide our final value by `N` to get the mean.

Profiling version 3 of the code (see Figure 7.5), most of the time is now spent calculating the means and standard deviations of the distance matrices, addressed in the next code version.

Version 4 In this version, we change the approach used for calculating the means and standard deviations of the distance matrices. First, we should note that the naive one-pass method (used originally for calculating the mean and standard deviation of the correlations) has a tendency to accumulate rounding errors when performed on a computer with very large numbers (and lots of numbers) – when a much smaller number is added to a very large number on a computer, the smaller number can appear to be "lost" in the final result. With this in mind, Welford's one-pass method [196] was chosen. It works recursively in order to calculate both the mean and the standard deviation in one pass. The pseudocode for this method is given below:

```
mean_current = array[0]
sum_current = 0
count = 1

for i = 1 to (array length - 1) {
count++

mean_previous = mean_current
mean_current = mean_previous +
(array[i] - mean_previous)/count
```

	Name	Launches	Device %	Total (µs)
1	calcCorrelationsNoRepeats	1	0.02	596.077
2	calcMeanSDDistMatrix	1	0.01	200.292
3	reduction_explicitloop	4	0.00	74.477
4	calcDistMatrixParallelPerColumn	1	0.00	58.155
5	origZScorePerElementAll	1	0.00	18.243
6	valueMinusMeanParallel	1	0.00	15.107

Figure 7.6: NSIGHT profiler information for Version 4 of the PlotCorr code.

	Name	Launches	Device %	Total (µs)
1	calcCorrelationsImprov	1	0.01	166.527
2	calcMeanSDDistMatrix	1	0.01	151.420
3	reduction_explicitloop	4	0.00	56.362
4	calcDistMatrixParallelPerColumn	1	0.00	43.945
5	origZScorePerElementAll	1	0.00	13.507
6	valueMinusMeanParallel	1	0.00	11.362

Figure 7.7: NSIGHT profiler information for Version 5 of the PlotCorr code.

```
sum_previous = sum_current
sum_current = sum_previous
+ (array[i] - mean_previous) * sum_previous
+ (array[i] - mean_current)
}

mean = mean_current
standardDeviation = sqrt(sum_current/count)
```

Each iteration depends on the one before. This method cannot be easily broken down and parallelised with each thread working on a small piece of the puzzle. However, as we wish to calculate the mean and standard deviation of each of the distance matrices, we can perform this method for each distance matrix in parallel with the others.

Profiling this code, we found that the function for calculating the correlations was now taking the most time (see Figure 7.6); this is addressed in the next, final code version.

Version	Time (seconds)
1	0.160675
2	0.174797
3	0.038742
4	0.000962
5	0.000443

Table 7.1: Time spent on computation for each version of PlotCorr. Times were calculated by summing all the time spent on functions.

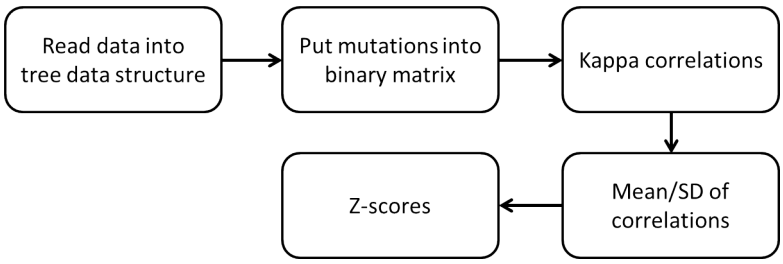

Figure 7.8: The functions involved in the Waddell – Kappa method.

Version 5 This final version of PlotCorr improves upon the correlations calculation function. As part of this calculation, we work out the average of the differences in distance matrix values; in the previous version of this code, all distance matrix values are examined; in the improved version, only half are examined, and the resulting sum is doubled to get the final sum used to calculate the average (the distance matrix is symmetric). In addition, as described in Section 2.9, where data is stored impacts the performance of accessing that data. With this in mind, we found improved performance by storing those pieces of data we needed to access multiple times in local variables (automatic variables stored in registers), and then accessing these variables directly. (All data is passed to the method in global memory, which has much lower performance than local variables.) Profiling the code a final time, we see the times given in Figure 7.7.

In this section we described how the PlotCorr method was parallelised in CUDA. Table 7.1 shows the time spent on computation in each version of the code. We shall now describe how the parametric methods were parallelised.

7.1.2 Waddell parametric methods

First, for each function within the Waddell – Kappa and Waddell – MI algorithms, we shall briefly describe what this function does (Figure 7.8 and Figure 7.9 show how these functions fit together to make the overall Waddell – Kappa and Waddell – MI methods). Then, more detail will be given on how the APOD cycle was used to improve the performance of these methods

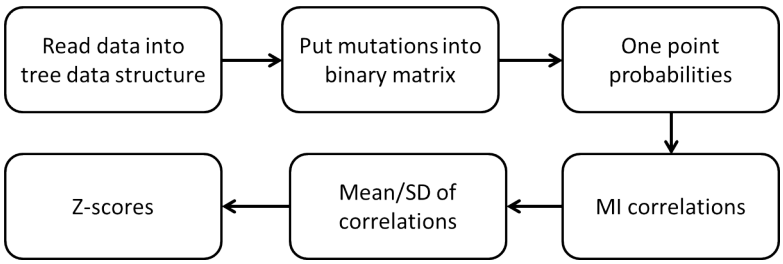

Figure 7.9: The functions involved in the Waddell – MI method.

overall. (The two parametric methods are described together as they share a lot of functions in common, as can be seen in Figure 7.8 and Figure 7.9.)

Method descriptions

The first step of both the Waddell – Kappa and Waddell – MI methods is to **read in the input tree and sequences into data structures usable by the program**. Bearing in mind that the next step of these methods depends on inspecting each branch (connection between nodes) independently of the others, we decided to create a "flat" tree data structure to represent the information, instead of the classic tree data structure. (This "classic" data structure involves node objects being connected by links.) The input Newick file (a text file describing the structure of the tree) is read into three arrays: `child1s[]`, `child2s[]`, and `seqs[]`. The `child1s[]` and `child2s[]` arrays tell us, for element i, which children derive from that node (if there are any). The `seqs[]` array tells us the sequence at each node.

Taking the arrays `child1s[]`, `child2s[]`, and `seqs[]`, we **work out "where" the mutations took place** on the branches of the tree. This involves inspecting the branches leading from each node in turn, and if, for example, at the beginning (top) of a branch there is a character A, and at the bottom of the branch is a different character B, we place a 1 in the binary matrix in the row corresponding to this branch, and the column corresponding to this position in the alignment. If the two characters are the same, we put a 0 in that position. This step is performed for both Waddell – MI and Waddell – Kappa.

There are two types of correlation we can calculate for our binary matrix (this is where the Waddell – Kappa and Waddell – MI differ).

Cohen's Kappa is a measure of the amount of inter-annotator agreement [33]. In our case, we are calculating, for each pair of columns in the binary matrix, the amount of "agreement" between the pairs of columns in that ma-

trix, i.e. how many times does a 0 or 1 appear in both columns in the same row ("agreements"), versus how many times we see a 0 in one column and a 1 in another ("disagreements").

The method of calculating **Mutual Information** (as an alternative to Cohen's Kappa) is identical to that given in Section 3.1.1, but the "alphabet" used, instead of being the 20 amino acid alphabet, is simply the set $\{0, 1\}$. First we calculate the **one point probability** for both of the characters at each of the columns (telling us how often 0 and 1 appear in each column). We then calculate the **Mutual Information** of each pair of columns in the alignment.

Finally, Z-scores are calculated for each of the pairs of columns as described for PlotCorr.

Each of the statements in bold corresponds to one or more functions in the code, and can be thought of as one coherent piece of the puzzle.

Application of APOD

In the previous section, the various functions which make up the Waddell – Kappa and Waddell – MI methods were briefly described. In this section, more detail will be given on how the APOD cycle was used to parallelise these two methods. It should be noted that the code for Waddell – Kappa and Waddell – MI did not iterate through code versions like PlotCorr did, due in part to the fact that the finalised Z-score code developed for PlotCorr was utilised for the parametric methods from their first versions. This meant that only one version of each method was developed.

Waddell – Kappa version 1 As with PlotCorr, the parametric versions were implemented in parallel as much as possible initially. Reading the input files into the appropriate data structures was done in serial. Tree traversal is the process of moving around a tree data structure which consists of nodes and connections between nodes. Methods have been developed in the past which aim to improve the performance of the traversal of tree data structures in parallel (this task is not normally well-adapted to parallel architectures). We did not consider such methods for our implementation due to the fact that we only need to traverse the tree once (inspecting each branch once).

Next we need to construct a binary matrix which captures the mutation information on the branches and allows for the calculation of correlations in the next step. For each node, the thread checks, for each of its (up to 2) children

	Name	Launches	Device %	Total (µs)
1	putMutationsInMatrixImprov_kernel	1	0.03	941.741
2	kappa_kernel	1	0.01	149.532
3	reduction_explicitloop	4	0.00	55.978
4	origZScorePerElementAll	1	0.00	13.347
5	valueMinusMeanParallel	1	0.00	11.522

Figure 7.10: NSIGHT profiler information for the Waddell – Kappa code.

	Name	Launches	Device %	Total (µs)
1	putMutationsInMatrixImprov_kernel	1	0.04	1,251.664
2	computeMI_kernel	1	0.01	288.727
3	reduction_explicitloop	4	0.00	74.157
4	origZScorePerElementAll	1	0.00	17.731
5	calculateOnePointProb_kernel	1	0.00	16.291
6	valueMinusMeanParallel	1	0.00	15.331

Figure 7.11: NSIGHT profiler information for the Waddell – MI code.

whether each of the characters in the current node equal the characters in the corresponding positions of the children, putting a 1 or 0 in the binary matrix as applicable.

Calculating the Kappa correlations themselves, the code is parallel over pairs of columns; that is, the correlation for each pair of columns is calculated by a different thread. Like with PlotCorr, the value of Cohen's Kappa for two positions i and j is equal to the Cohen's Kappa for j and i, so we use arrays of indices to keep track of the pair that each thread is calculating Kappa for.

Finally, the code for calculating Z-scores is the same as that used for PlotCorr.

The information produced from profiling the code can be seen in Figure 7.10.

Waddell – MI version 1 The code for all methods except the correlations is the same as that for Waddell – Kappa. For the Waddell – MI code, each thread calculates the one point probabilities for its own column. Also, like with our PlotCorr correlations code (and our Waddell – Kappa code), we use two arrays, `index1` and `index2`, to work out which pair of columns we want

this thread to work on for the MI values, so that we can focus on computing only the correlations necessary. The information produced from profiling the code can be seen in Figure 7.11.

7.2 Full-program method times

In the previous section the approach used to parallelise the methods was described, as well as how this approach was applied. In this section, the experimental set-up used, and the approach taken for timing the methods, will be described. Then the results of timing the methods in serial (C++), in CUDA C on a NVIDIA GeForce GTX 660 Ti, and a NVIDIA GeForce Titan X (Pascal) will be presented.

7.2.1 Experimental set-up

Serial For the serial timings, we used a desktop computer running Linux Ubuntu 16.04.3, with an Intel Core i7-3770T CPU (2.50 GHz) CPU. The compiler used was g++ (the version given as gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1 16.04.4)).

Parallel For the parallel timings, we used two separate desktop PCs; both of these were running Linux Ubuntu 16.04.3. The first had a NVIDIA GTX 660 Ti, the other a Titan X (Pascal). The version of CUDA in both cases is 7.5, and the driver installed in both cases is 370.28.

7.2.2 Approach taken to timing

We took whole-program times using `clock_gettime()` with the argument specifying a monotonic clock [39]. We chose to use this timing method because if we use it to take a time at one point in the program, and then again later on, it is guaranteed that the second time will not be less than the first (this is the monotonic characteristic). The monotonic clock is also unaffected by alterations/corrections to the real-time clock. We used this for the CUDA implementation as well as the serial one because the total time of the CUDA code depends on serial code, and we wanted to keep the method used for timing consistent between the different implementations.

The alignment size parameters were varied as follows:

- Keeping the alignment length constant (at 64), vary the number of sequences with the values 2, 10, 18, 25, 33, 41, 49, 57, 64, 73, 80

- Keeping the number of sequences constant (at 2), vary alignment length with the values 51, 925, 1799, 2673, 3547, 4422, 5296, 6170, 7044, 7918, 8792

The maximum and minimum values of the alignment lengths and number of sequences varied correspond to the real data's maximum and minimum alignment size parameters. The constant value of alignment length was chosen by simply identifying the alignment with 80 sequences, and using that length (the alignments with 2, 10, 18, etc. sequences were constructed by removing sequences from the alignment with 80 sequences). A similar approach was used for keeping the number of sequences constant (the alignment with length 8792 had 2 sequences).

7.2.3 Timing results

We present the results of each co-evolution detection method separately, on each hardware set-up (CPU, GTX 660 Ti, Titan X), varying the alignment length and number of sequences individually.

PlotCorr

Varying the alignment length, all three sets of hardware timings (CPU, GTX 660 Ti, and Titan X) are plotted in Figure 7.12. Surprisingly we can see that overall, the Titan X has the worst performance initially, but that at the longest alignment length, the serial implementation takes as much time as the GTX 660 Ti.

Varying the number of sequences (see Figure 7.13), we see that, again the Titan X seems to give the worst performance, and serial the best, but just from observing the slopes of the different methods on the graph, it is plausible that the CPU implementation may overall have the worst performance. Indeed, repeating the timing experiments with a longer base alignment length (574), we obtain the results in Figure 7.14. For the smallest numbers of sequences, serial is best, but for the overwhelming majority of numbers of sequences, the two parallel implementations take the shortest amount of time. We also see that there is some overlap between the GTX 660 Ti and the Titan X times obtained. We can easily see that the serial implementation is most affected by the number of sequences in the alignment.

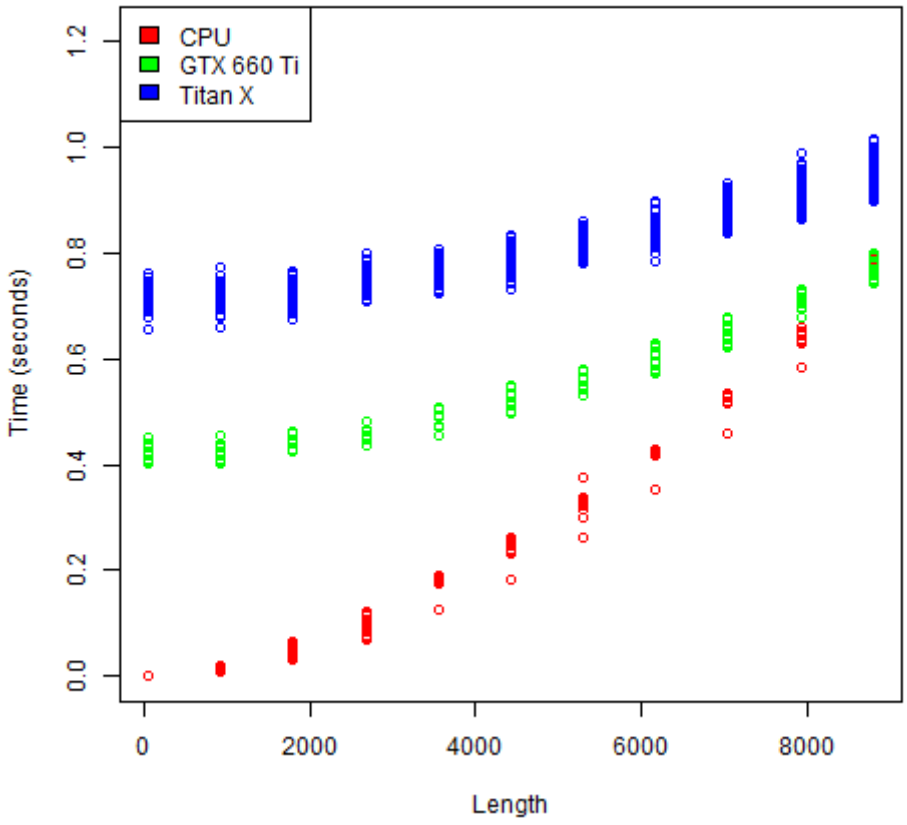

Figure 7.12: PlotCorr whole-program times; each point is an individual run; the alignment length is varied while the number of sequences is kept constant.

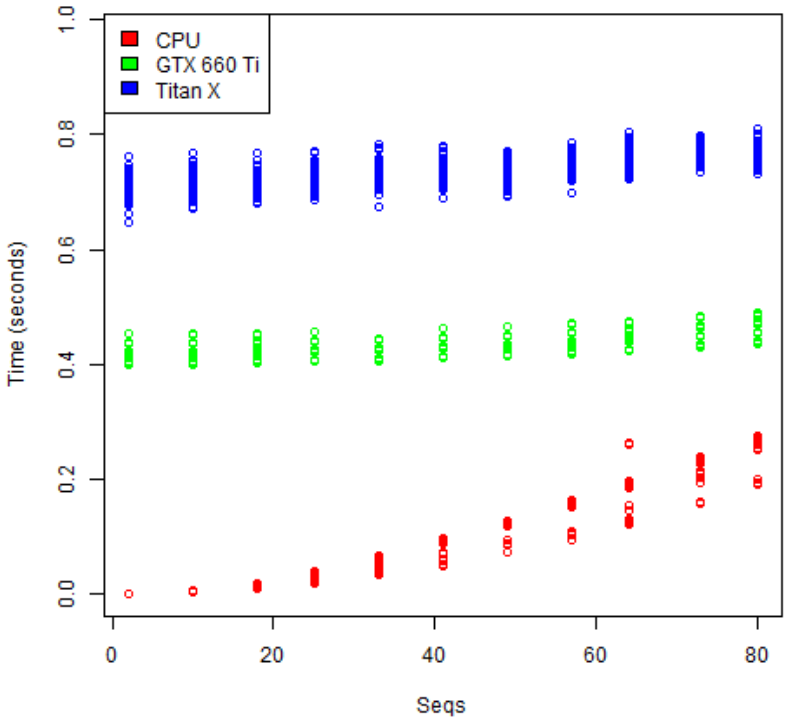

Figure 7.13: PlotCorr whole-program times; each point is an individual run; the number of sequences is varied while the alignment length is kept constant.

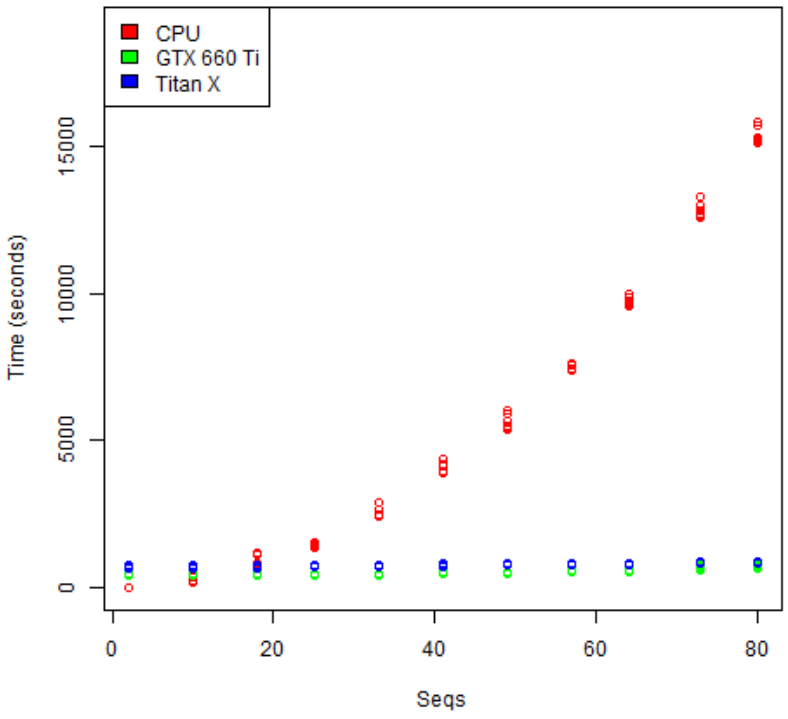

Figure 7.14: PlotCorr whole-program times; each point is an individual run; the number of sequences is varied while the alignment length is kept constant. These timings are for an increased alignment length of 574.

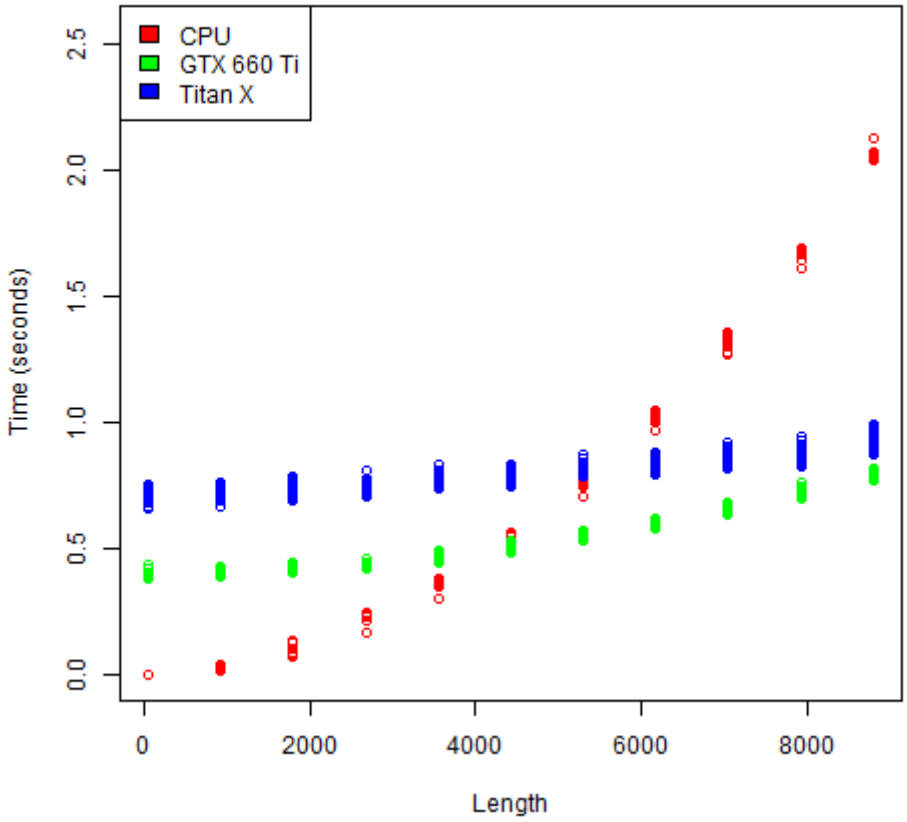

Figure 7.15: Waddell – Kappa whole-program times; each point is an individual run; the alignment length is varied while the number of sequences is kept constant.

Waddell – Kappa

Varying the alignment length, Figure 7.15 shows outright that the serial method has the worst performance, and that there is some overlap between the GTX 660 Ti and the Titan X times for the longest alignment length. The serial implementation is therefore affected the worst by an increase in alignment length.

Varying the number of sequences, the graph for all three hardware set-ups is shown in Figure 7.16. We can see that the serial implementation performs the best, but when we increase the alignment length to 8792, we can see that the serial implementation performs worst, and there is some overlap between the two sets of parallel times (see Figure 7.17). Again, the serial implementation is worst affected.

Waddell – MI

Varying the length, we see a similar pattern to that seen with Waddell – Kappa (see Figure 7.18), where the serial implementation is obviously the worst-performing, and there is overlap between the GTX 660 Ti and the Titan X.

Varying the number of sequences, the graph showing the timings for all three hardware set-ups is within Figure 7.19. It is as we saw for Waddell – Kappa. Increasing the base alignment length to 8792, we can see that not only does the serial implementation perform the worst, there is, again, overlap between the times on the two graphics cards.

Overall, we find that for sufficiently large data sets, the GTX 660 Ti and the Titan X perform similarly, and the serial implementation performs the worst. We believe we need sufficiently large datasets to see this effect due to the overhead required to run programs in parallel – data needs to first be copied to the GPU, and also copied back after computation has been completed (we included these timings in the analysis). Also, it should be noted that PlotCorr does not require such long alignments as the two parametric methods in order for the parallel methods to perform the best. Generally, the serial implementation seems to be more affected by alignment size parameters than the parallel methods.

Now that we have shown the overall timing results, we perform a cost/benefit analysis between these hardware systems and a supercomputer.

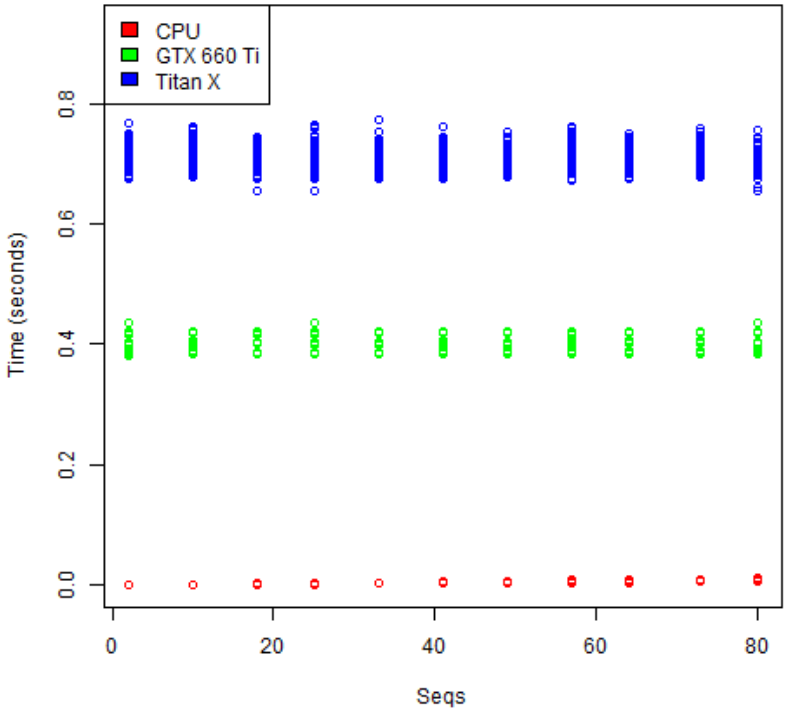

Figure 7.16: Waddell – Kappa whole-program times; each point is an individual run; the number of sequences is varied while the alignment length is kept constant.

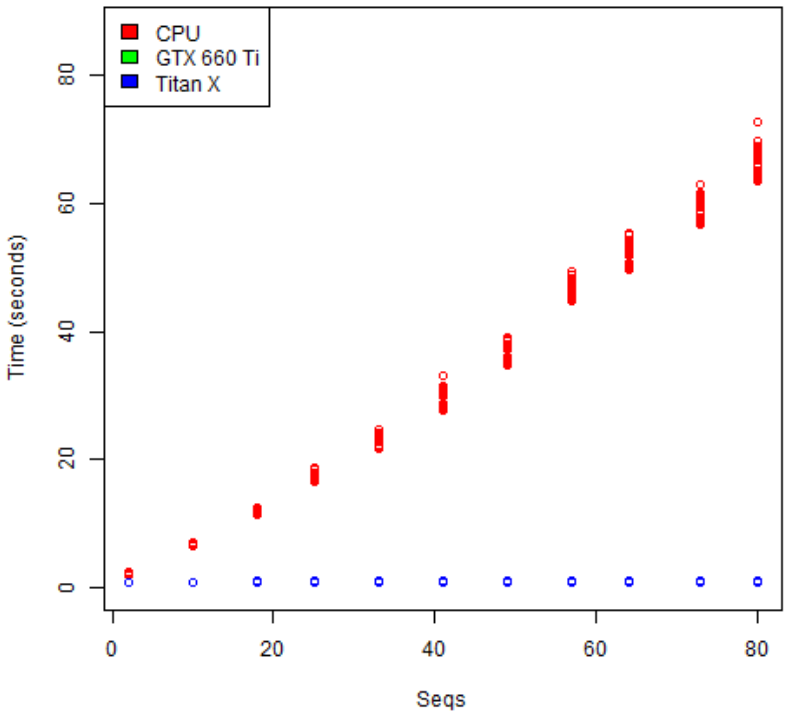

Figure 7.17: Waddell – Kappa whole-program times; each point is an individual run; the number of sequences is varied while the alignment length is kept constant. The base alignment length is increased to 8792.

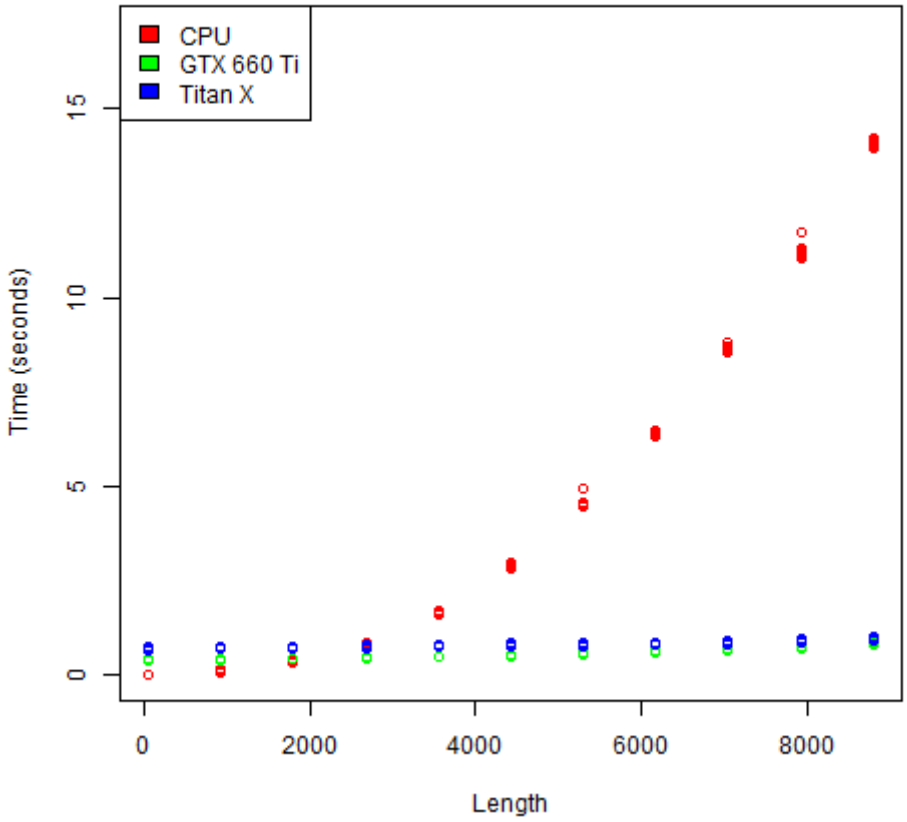

Figure 7.18: Waddell – MI whole-program times; each point is an individual run; the alignment length is varied while the number of sequences is kept constant.

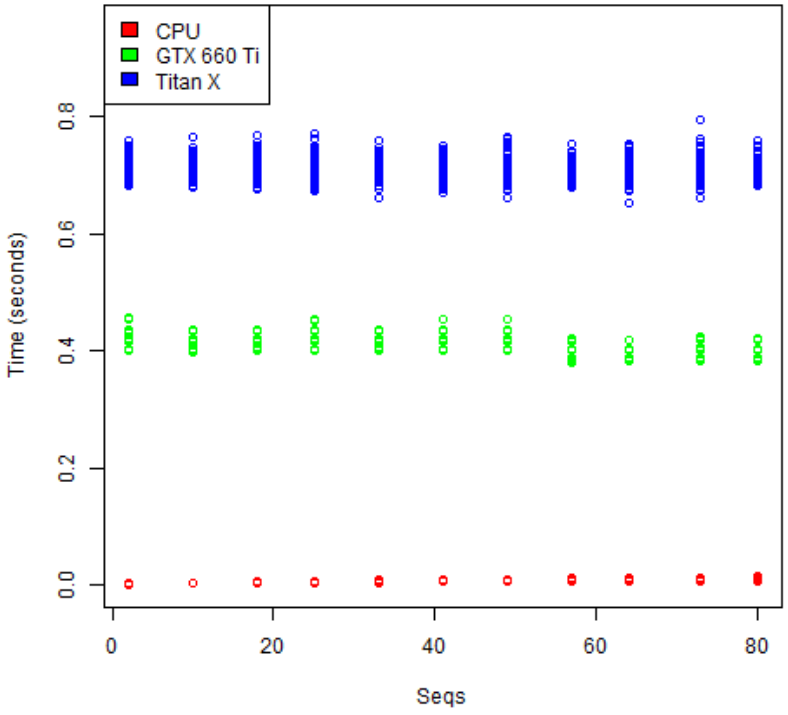

Figure 7.19: Waddell – MI whole-program times; each point is an individual run; the number of sequences is varied while the alignment length is kept constant.

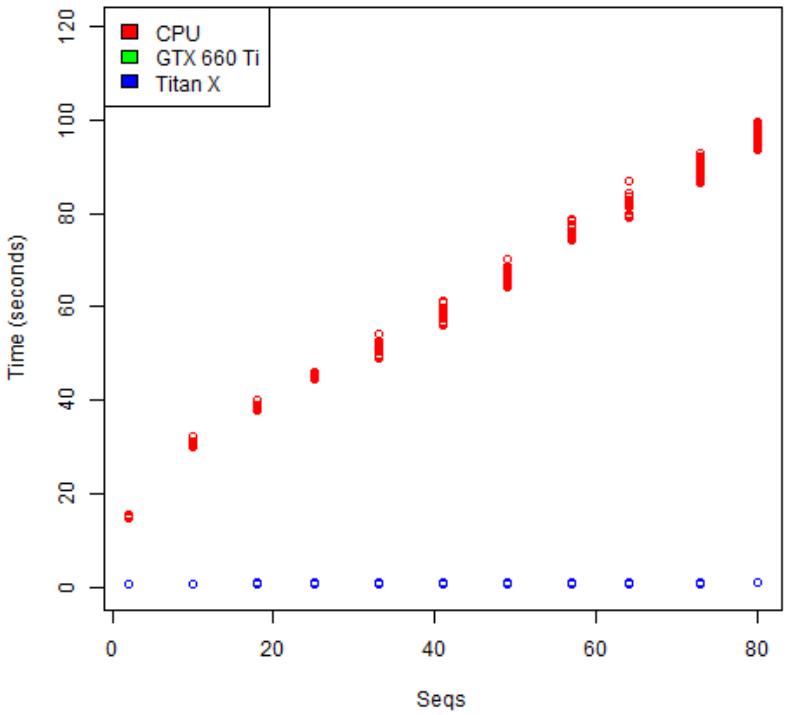

Figure 7.20: Waddell – MI whole-program times; each point is an individual run; the number of sequences is varied while the alignment length is kept constant. The base alignment length is increased to 8792.

7.3 Cost/benefit analysis

In this section, we compare the use of a CPU-only system, a supercomputer (Raven), as well as the two graphics cards used, in terms of a cost-benefit analysis. We compare the number of "average-sized jobs" that could be run in a 28-day month, defining these jobs as running PlotCorr on an alignment of 6 sequences with an alignment length of 574 (the average number of sequences and average alignment length for the real data set). We then repeat these calculations for a larger alignment, finding that this gives better performance on the GPUs, and confirming that large-enough alignments are necessary for running a job in parallel to be worth doing.

7.3.1 CPU-only system

The CPU used is an Intel Core i7-3770T CPU (2.50 GHz) CPU. The average time taken to run an "average-sized job" is 0.1242865473 seconds, which allows a maximum of 2419200 / 0.1242865473 $\approx$ 19464697 jobs in a 28-day month.

7.3.2 Raven

Raven is ARCCA (Advanced Research Computing Cardiff)'s supercomputer. For the purposes of our project, we ran our serial code on the serial nodes. It costs (in August 2017) 2 pence per CPU core hour to run code on a serial node. Always running code on one core, this equates to 2p per hour. Each user has the ability to run a maximum of 20 jobs at any one time. The average time taken to run an "average-sized job" is 0.3142269054 seconds, which equates to 2419200 / 0.3142269054 $\approx$ 7698895 jobs on a single serial node, or 153977900 total jobs using the maximum 20 consecutive jobs available to the average Raven user. To run 20 consecutive "average-sized" jobs constantly throughout a 28-day month (running another job once one is completed) would cost 20 jobs * (24 hours * 28 days) * £0.02 = £268.80.

7.3.3 GeForce GTX 660 Ti

Around a month before this was purchased in 2013, its price was listed on one website as £174.79 [164]. It has since (officially) been discontinued, although at time of writing we managed to find the graphics card being sold by a limited number of e-retailers (prices listed include VAT):

- Produced by MSI; GTX 660 Ti Power Edition £230.39 [149]
- Produced By Zotac; GTX 660 Ti £199.19 [150]

Hardware	Jobs	Cost
CPU	19,464,697	unknown
Raven	153,977,900	£268.80
GTX 660 Ti	5,696,638	£211.19
Titan X	3,401,378	£1099

Table 7.2: The number of "average-sized" jobs that could be run in a 28-day month, and the cost of doing so, for several hardware options. We could not find a cost for the CPU used.

- Produced by Gigabyte; GTX 660 Ti £203.99 [148]

We take the price of this graphics card to be a one-off cost, i.e. the price cost of this method is (an average of) £211.19.

The average time taken to run an "average-sized job" is 0.4246715045 seconds, which allows a maximum of 2419200 / 0.4246715045 $\approx$ 5696638 jobs in a 28-day month.

7.3.4 GeForce GTX Titan X

This graphics card was purchased a matter of weeks after its release; it cost £1099 [137].

The average time taken to run an "average-sized job" is 0.711240972 seconds, which allows a maximum of 2419200 / 0.711240972 $\approx$ 3401378 jobs in a 28-day month.

Table 7.2 summarises the findings of the initial cost/benefit analysis using "average-sized" jobs. What is interesting to note is that Raven can run the largest number of jobs for the second-cheapest price, and the Titan X is the most expensive option, while running the smallest number of jobs in a month.

7.3.5 Larger alignments

As was found earlier within this chapter, larger alignments can show that the parallel implementations perform better overall than smaller ones, so we re-run the calculations for an alignment with 80 sequences, and of length 1536. The findings of this cost/benefit analysis are summarised below:

- CPU = 111.308 seconds – maximum of 2419200 / 111.308 $\approx$ 21734 jobs
- Raven = 215.766 seconds – maximum of (2419200 / 215.766) * 20 $\approx$ 224242 jobs

- GTX 660 Ti = 4.371733089 seconds – maximum of 2419200 / 4.371733089 $\approx$ 553373 jobs
- Titan X = 1.659154726 seconds – maximum of 2419200 / 1.659154726 $\approx$ 1458091 jobs

For larger alignments, what is interesting to note is that Raven takes longer to run one job than the CPU does; it only achieves better performance through virtue of being able to run 20 jobs consecutively. Overall, the Titan X can run 67 times as many jobs as on a CPU-only system, or 6.5 times as many as on Raven. To run as many jobs on Raven as the Titan X can run in a month, it would take 215.766 seconds * 1458091 jobs = 314606462.706 seconds, which is 5243441.0451 minutes, or 87390.684085 hours (130 months or almost 11 years). It is not necessary to run costing calculations here; the Titan X clearly works out better for larger alignments.

One caveat to this cost/benefit analysis is that only a subset of alignments of proteins within the human proteome will be this large, but at the same time, this analysis shows how, as more and more sequences become available, the improved performance of GPUs can be harnessed.

7.4 Summary

In this chapter the approach used to parallelise the three co-evolution detection methods (PlotCorr, Waddell – Kappa, and Waddell – MI) was described, and the results of timing these methods were presented. It was found that, as expected, the parallelised methods performed better than the serial methods, as long as sufficiently large alignments were used. In addition, a general cost/benefit analysis was presented, comparing the graphics cards used against a supercomputer and a CPU-only system, showing that graphics cards provide a better deal in the long-run.

Chapter 8

Application of Methods to Real Data

In the previous chapter, we discussed how we parallelised PlotCorr and the two parametric methods using CUDA. We also compared the performance of these CUDA implementations to serial implementations of the same methods. As discussed previously, before parallelisation it would have taken a prohibitively long time to run (serial) co-evolution detection methods on the entire human proteome. Through the process of parallelisation we can now perform an analysis of the entire human proteome to find co-evolving sites, which we shall present in this chapter.

First we shall overview some of the alternative methods for identifying sites important to a protein's function, at which mutations are more likely to be deleterious. Then we shall detail the results of the analysis of our real data, using the parallel versions of PlotCorr, Waddell – MI, and Waddell – Kappa, comparing these methods to the alternatives already in use. The step this chapter covers is shown in Figure 8.1.

8.1 Alternative methods for classifying mutations

In this section we shall overview methods for classifying mutations as damaging or benign. The ones we shall focus on are SIFT and PolyPhen, as these are widely used in the field [123].

8.1.1 SIFT

SIFT, which Sorts Intolerant From Tolerant mutations, works by calculating the probability of each possible mutation at every position in an input alignment. If the calculated probability of a mutation is greater than a given

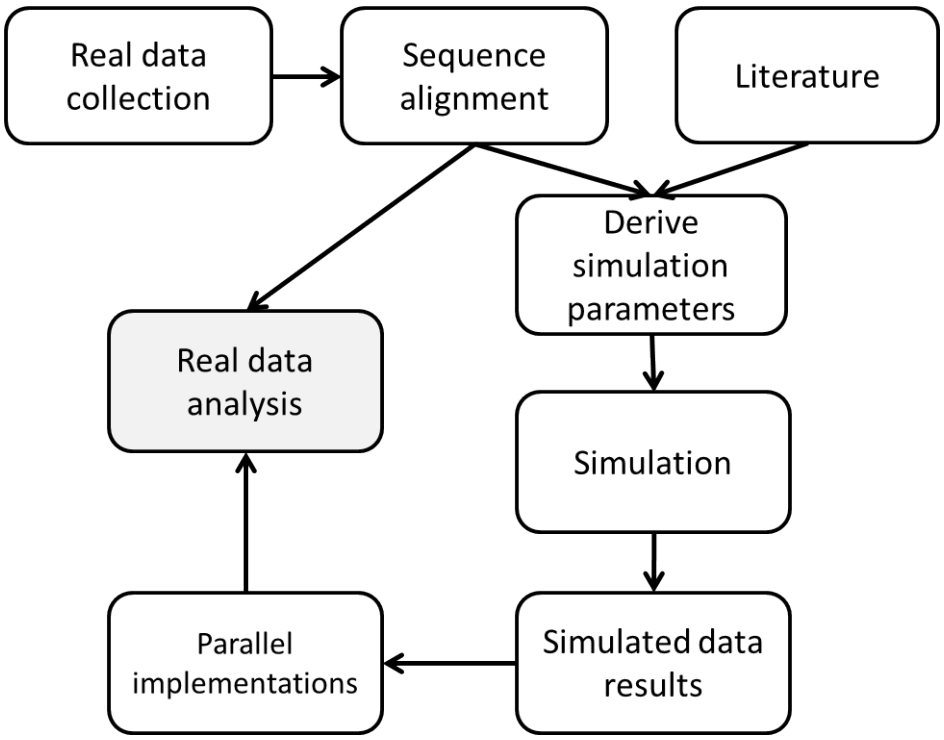

Figure 8.1: This chapter covers analysis of the real data alignments using the parallel methods implemented in the previous chapter (shaded box).

threshold, it is classified as tolerated; if not, then it is classified as not tolerated [132]. SIFT's calculations take into account the frequencies of amino acids in each column of an alignment, as well as how conserved the column is; the more conserved a column is, the less likely it should be able to tolerate mutations.

A version of SIFT called SIFT Indel [92] allows the user to predict whether insertions or deletions (of a multiple of 3 nucleotides in length) are tolerated or not. This program takes into account the physicochemical properties of the amino acids that are being inserted/deleted, whether the amino acids are in a particular type of secondary structure, as well as other characteristics of the data. The three classification rules found to predict most training mutations were to do with whether or not the insertion/deletion occurs in a Pfam functional domain[1], whether it is a repeat or not, resides in a disordered region (a region of the protein without a defined three-dimensional structure [12]), or if the position to the left of the insertion/deletion is conserved or not.

SIFT Indel was originally developed to classify frame-shifting insertions/deletions (i.e. those which are not divisible by three, and therefore completely change the amino acid sequence following the mutation by shifting the

[1]Pfam is a protein database; a functional domain is a distinct unit within the protein responsible for a particular function [163]

start position of subsequent codons) [91]. An example is that of the gene sequence ACGTCGGAG changing to ATCGGAG; when before, the first amino acid came from the codon ACG (T), it is now ATC (I). The original version of SIFT Indel found four features that gave the best performance; these were to do with the number of conserved nucleotides and the location of the insertion/deletion (indel) in the gene.

8.1.2 PolyPhen

PolyPhen, which stands for Polymorphism Phenotyping, is a web server for classifying non-synonymous SNPs (Single Nucleotide Polymorphisms, where there are multiple common variants of a single nucleotide present in a population) in coding sequences as to whether they are deleterious or not [160]. The input taken is a protein sequence (or an ID linking to the protein in the SWALL database (this was also called SPTR (Swall) [15], and appears to have been simply the complete database constituting the union of the separate Swiss-Prot and TrEMBL sequence databases [53] (now wholly contained within UniProt [186])), and the two amino acid variants characterising the polymorphism (i.e. the two variants of an amino acid present in a population, at a particular position in the proteome). It should be noted that a polymorphism refers to a common variant, whereas in this thesis, we are primarily interested in rare variants.

PolyPhen obtains annotation information from the Feature Table of the SWALL database, and also uses various programs to predict whether the amino acid variant occurs in a transmembrane region, a coiled coil region (these are simply specific types of structures or regions within a protein), etc. It then obtains sequences that are ancestrally related to the input sequences (homologues), and runs the PSIC (Position-Specific Independent Counts) program on these sequences to calculate the profile matrix. The values within the profile matrix are 'logarithmic ratios of the likelihood of a given amino acid occurring at a particular position to the likelihood of this amino acid occurring at any position' [3, pp. 25]. The difference between the PSIC scores of the two amino acid variants is calculated, and if this difference is particularly large, this means that the substitution in question is rarely, if ever, seen in the protein being considered. Different values related to secondary structure, etc. are then calculated/obtained. Contacts are found in the 3D structure of the protein, between the position we are focussing on and: ligands, subunits of the protein we are looking at, and other important positions.

The three types of data collected in the previous step (feature information, profile matrix, secondary structure information) are used in the following step, which involves following a set of rules to obtain a prediction as to whether the amino acid variant is deleterious or not. The rules relate to things such as whether the site is known to be important to function, relatively close in space to the active site, etc., and the variant is classified as either damaging, benign, or N/A. The rules were chosen based on a comparison of proposed rules on datasets of mutations associated with disease, nsSNPs (non-synonymous SNPs, or SNPs that cause a change to the corresponding amino acid) and substitutions between human sequences and orthologous sequences from mammals closely evolutionarily related to humans.

In the paper by Adzhubei et al. [4] PolyPhen-2 was introduced. In this updated version of the program, there are different features, a different approach to creating alignments, and a different way of classifying the amino acid variant (a naive Bayes classifier, based on Bayes' Theorem, is now used). There are two versions of PolyPhen-2, depending on which dataset it was trained on. The version trained on a dataset called HumVar should be used for distinguishing between those mutations that have large effects on health from normal human variation, while the version trained on HumDiv is meant to be used on rare variations at genomic positions associated with complex diseases.

Now that we have reviewed SIFT and PolyPhen, we shall now consider the analysis of the results of performing the co-evolution detection methods on the entire human proteome.

8.2 Real data analysis

Having run the final parallel PlotCorr and Waddell programs on the real data collected previously (a set of alignments constructed from the proteins in the human proteome along with related species), we have a set of results for the PlotCorr algorithm, a set of results for Waddell – MI, and a set of results for Waddell – Kappa. In this section, we begin by reviewing the bias towards certain amino acids in the dataset. We then introduce the Fromer et al. [66] dataset, which we shall also be using, list the hypotheses we shall be testing, and actually test them.

8.2.1 Amino acid frequencies

We begin by investigating whether certain amino acids are more likely to appear at co-evolving sites than others. Figure 8.2, Figure 8.3, and Figure 8.4 show the frequencies of each amino acid in all sites (red bars) with the frequencies in predicted co-evolving sites (overlaid in green bars) for PlotCorr, Waddell – Kappa, and Waddell – MI, respectively, with a low unique amino acid (UAA) threshold. It is interesting to note that in all three cases, most of the amino acid frequencies in co-evolving sites are significantly different at the 0.05 level (indicated by an asterisk above the appropriate bar) to the frequencies in all sites. For each of the graphs, each of the amino acids was tested by running a logistic regression using the model:

$$\text{status} \sim \text{amino acid} \tag{8.1}$$

where "status" is a binary value based on whether the position is co-evolving/ fully conserved or not, and "amino acid" is a binary value based on whether the appropriate amino acid at this position is the amino acid X or not (where X is one of the 20 amino acids; we run one analysis per amino acid). Across all 6 scores, occurrences of amino acids A and N are statistically significantly increased in predicted co-evolving sites relative to all sites, and L is statistically significantly decreased in predicted co-evolving sites relative to all sites.

The same can be seen for the higher UAA threshold (see Figure 8.5, Figure 8.6, and Figure 8.7 for PlotCorr, Waddell – Kappa, and Waddell – MI respectively.)

Now looking at conserved sites only, we find that there is still some disparity between the amino acids in all sites, and those in conserved sites (see Figure 8.8). Like with the co-evolving sites, most of the amino acids show significant differences in the frequencies being considered. What is interesting to note here is that the amino acids whose frequencies in detected co-evolving sites were consistently different to the frequencies in all sites show the opposite trend in conserved sites. For example, L is statistically significantly decreased in predicted co-evolving sites relative to all sites (across all 6 measures), but for conserved sites, L is significantly increased relative to all sites.

All of the p-values for the logarithmic model between co-evolution or conservation and amino acids can be found in Appendix A.11.

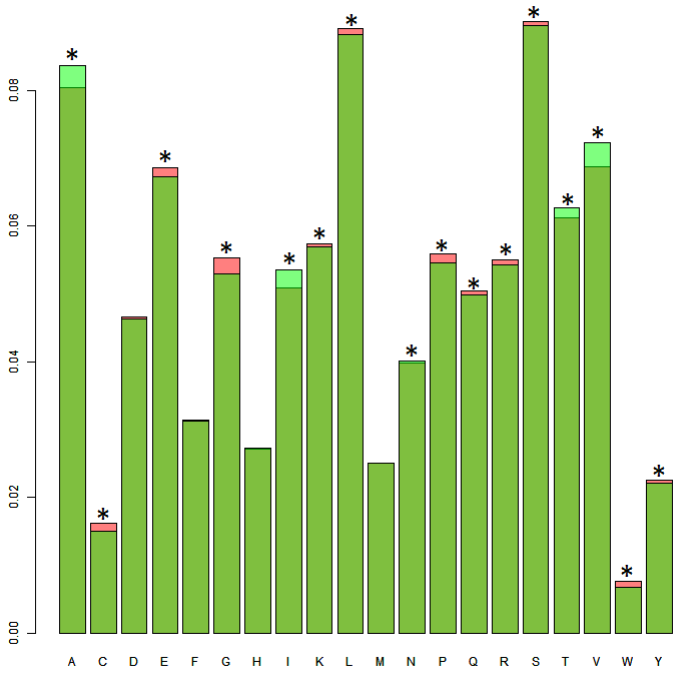

Figure 8.2: The frequencies (y axis) of the 20 amino acids (x axis) are different between the co-evolving sites identified by PlotCorr with a low unique amino acid threshold and all sites. Conserved sites are not considered. The frequencies in all sites are given by red bars, and then the frequencies in co-evolving sites identified by PlotCorr with lower UAA threshold are overlaid in green bars. This means that the dark green portions are where the bars overlap, and the colour at the top of the bars tells us which has the higher frequency.

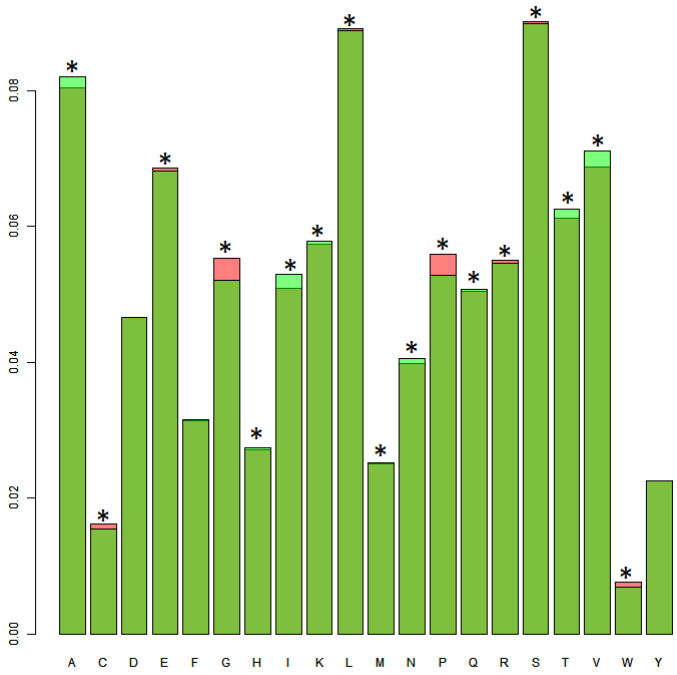

Figure 8.3: The frequencies (y axis) of the 20 amino acids (x axis) are different between the co-evolving sites identified by Waddell – Kappa with a low unique amino acid threshold and all sites. The rest of the description is the same as for Figure 8.2.

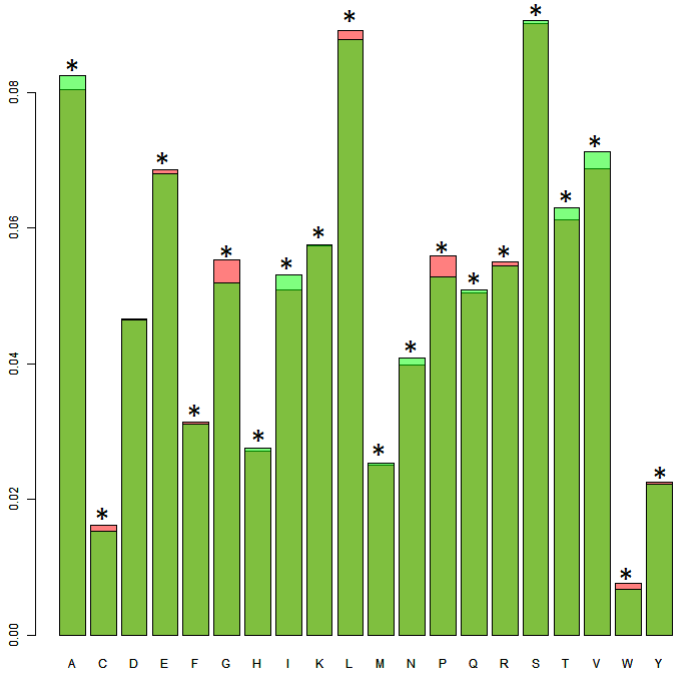

Figure 8.4: The frequencies (y axis) of the 20 amino acids (x axis) are different between the co-evolving sites identified by Waddell – MI with a low unique amino acid threshold and all sites. The rest of the description is the same as for Figure 8.2.

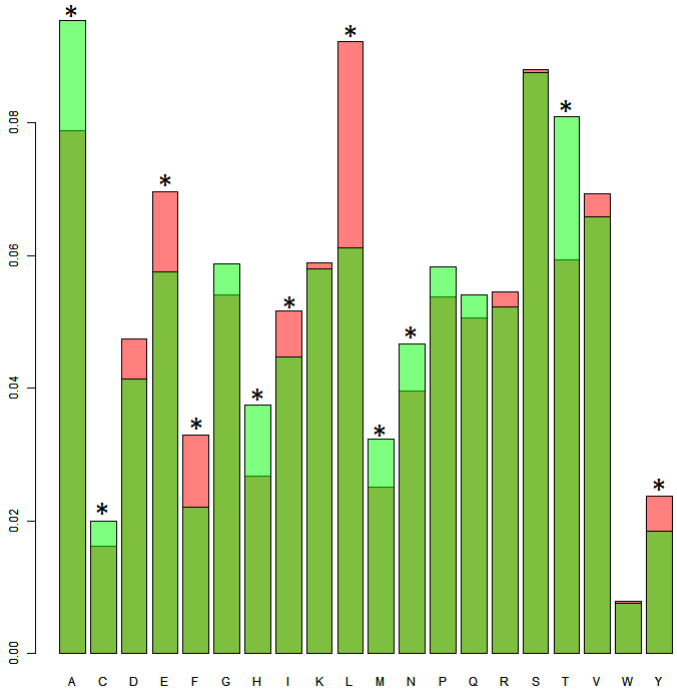

Figure 8.5: The frequencies (y axis) of the 20 amino acids (x axis) are different between the co-evolving sites identified by PlotCorr with a high unique amino acid threshold and all sites. Conserved sites are not considered. The frequencies in all sites are given by red bars, and then the frequencies in co-evolving sites identified by PlotCorr with higher UAA threshold are overlaid in green bars. This means that the dark green portions are where the bars overlap, and the colour at the top of the bars tells us which has the higher frequency.

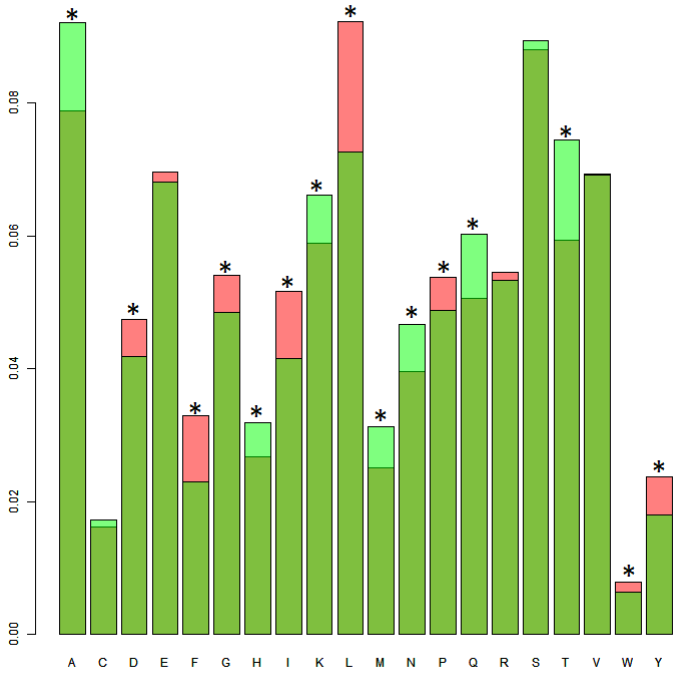

Figure 8.6: The frequencies (y axis) of the 20 amino acids (x axis) are different between the co-evolving sites identified by Waddell – Kappa with a high unique amino acid threshold and all sites. The rest of the description is the same as for Figure 8.5.

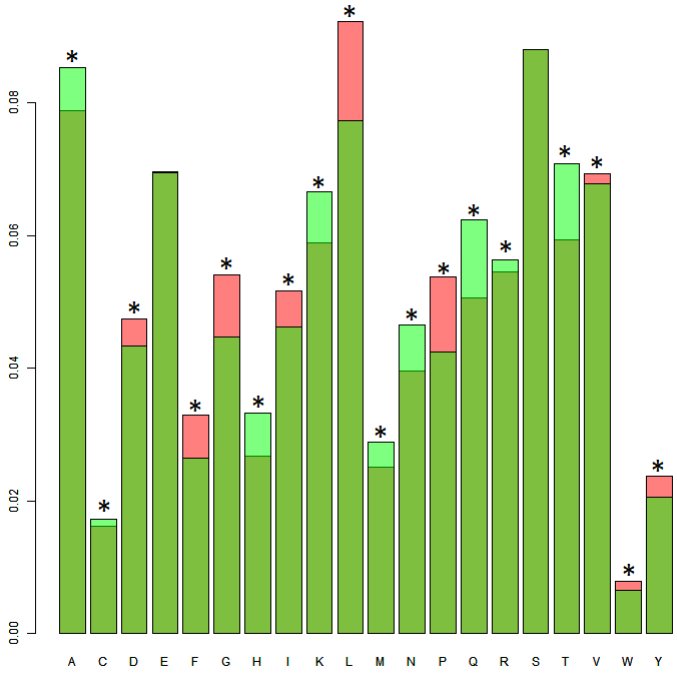

Figure 8.7: The frequencies (y axis) of the 20 amino acids (x axis) are different between the co-evolving sites identified by Waddell – MI with a high unique amino acid threshold and all sites. The rest of the description is the same as for Figure 8.5.

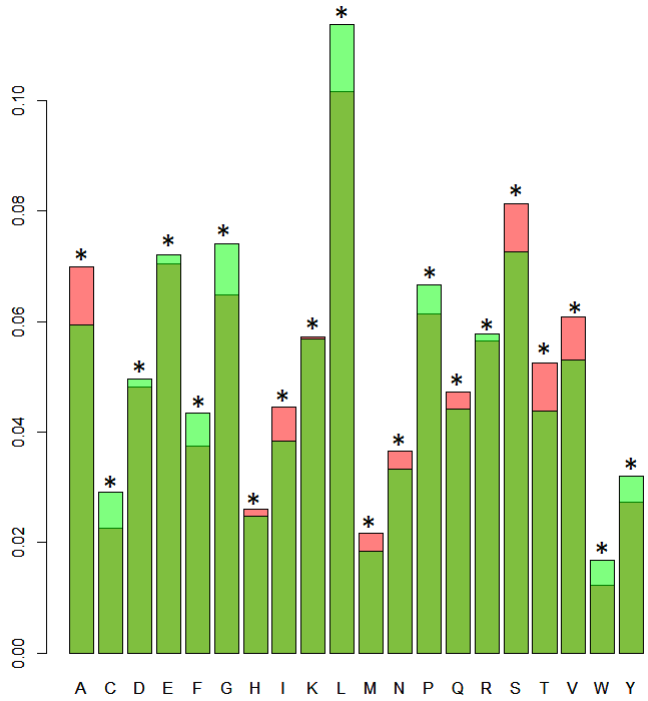

Figure 8.8: The frequencies (y axis) of the 20 amino acids (x axis) are different between the conserved sites and all sites. The frequencies in all sites are given by red bars, and then the frequencies in conserved sites are overlaid in green bars. This means that the dark green portions are where the bars overlap, and the colour at the top of the bars tells us which has the higher frequency.

8.2.2 Introduction to the data

As a reminder to the reader, the real data set collected in Chapter 4 consists of a set of multiple sequence alignments, each alignment being constructed from a particular human protein, along with homologous sequences from other species. In addition to this dataset, we also obtained a set of mutations present in cases (individuals with autism, intellectual disability, or schizophrenia) and controls from the Fromer et al. study [66]. The dataset consisted of data from an exome-sequencing study in cases with schizophrenia (generated specifically for that study), plus data from other studies of cases with schizophrenia, autism, and intellectual disability, as well as controls. There were a total of 2161 cases and 731 controls. The cases can be divided into 1043 with autism spectrum disorder, 151 with intellectual disability, and 967 with schizophrenia. The mutations in this data set are divided into silent (synonymous) and missense (non-synonymous) mutations. There are 681 missense and 223 silent mutations for the cases dataset, and 193 missense and 70 silent mutations for the controls dataset. All of the mutations in the entire dataset are de novo mutations; mutations present in the individual, but not in their parents. Contained within this dataset are SIFT and PolyPhen2 scores (the variant which used the HumDiv dataset) for all mutations. The main findings of the Fromer et al. study were:

1. There was not an increased rate of de novo mutations in cases compared to controls

2. Disease-causing mutations were enriched (over-represented) in genes relating to synaptic function (synapses are the points where impulses are passed between neurons [122])

3. There is commonality in the genes hit by loss-of-function mutations in the schizophrenia dataset and in autism spectrum disorder, and between schizophrenia and intellectual disability

In terms of severity of brain functional impairment (roughly speaking), intellectual disability (ID) is associated with more impairment than autism spectrum disorder (ASD), which has more impairment than schizophrenia (SCZ). Another finding of the Fromer et al. study, separate to these but relevant to the present thesis, is as follows. In terms of the ratio of the number of loss-of-function mutations (the most deleterious class of mutation, in terms of protein function) to number of missense mutations in the schizophrenia, autism spectrum disorder, and intellectual disability cases as separate groups, the largest ratio is for intellectual disability, then autism spectrum disorder,

then schizophrenia. This same order was found when looking at conservation scores of non-synonymous mutations.

Now that we have introduced the data we are working with, we shall introduce the hypotheses that we shall test using this data.

8.2.3 Data analysis introduction

This is a preliminary investigation into the hypothesis: measures of co-evolution can be used to identify functionally important sites that by their very nature cannot be found by looking at evolutionary conservation. We expect to find that:

1. Co-evolution score is negatively correlated with site conservation.
2. Since SIFT and PolyPhen use conservation as one of the measures informing predicted mutation severity, there should be positive correlation between SIFT/PolyPhen mutation severity predictions and site conservation. Consequently, this should be reflected in a relationship between co-evolution scores and SIFT/PolyPhen.

If co-evolution measures do indeed identify functional sites, then this may potentially be reflected in:

3. Non-synonymous (NS) mutations contributing to disease occurring at sites with higher co-evolution scores than:
   (a) Silent mutations
   (b) NS mutations that do not contribute to disease
4. Difference in NS mutation co-evolution scores between disorders, possibly reflecting differences in disease severity. As was detailed above, in the Fromer et al. [66] study, the burden of loss-of-function mutations was greatest in intellectual disability, followed by autism spectrum disorder, and then schizophrenia. This means we might expect to see the burden of less deleterious mutations in the opposite order, so schizophrenia, then autism spectrum disorder, then intellectual disability, with this possibly reflected in co-evolution scores.

We shall now test each of these hypotheses in turn.

Co-evolution scores and conservation

As discussed earlier in this chapter, SIFT and PolyPhen are two methods for classifying mutations as disease-causing or otherwise. SIFT scores $<=$ 0.05 are interpreted as damaging (D), whereas scores higher than 0.05 are interpreted as tolerated (T) [105]. In contrast, for PolyPhen, smaller values are interpreted as benign, and larger values as damaging. In PolyPhen, there are three possible predictions: probably damaging, possibly damaging, and benign [3]. To work out which classification applies, threshold false positive rates are used. (These are separate to the actual PolyPhen scores.) For the PolyPhen score considered in this thesis (that which uses the HumDiv data), the threshold pair is 5% and 10%. The false positive rate (FPR) corresponding to a mutation's posterior probability score is compared against each of the thresholds in the pair. If the FPR is less than or equal to the first threshold (5%), the mutation is predicted to be probably damaging. If the FPR is less than or equal to the second threshold (10%), then the mutation is predicted to be possibly damaging. If the FPR is above the second threshold, then the mutation is predicted to be benign.

One measure that both SIFT and PolyPhen take into account is site conservation (if a site is fully conserved, then that means that there is only one unique amino acid at that site). The conservation score used in the Fromer et al. dataset is GERP_RS, taken from the GERP program [34].

We compare the three co-evolution detection methods PlotCorr, Waddell – Kappa, and Waddell – MI to the GERP conservation score. We do so by calculating the correlations between the maximum Z-score (for each detection method) at each mutation's position with the GERP conservation score for that position:

- PlotCorr = -0.2759158
- Waddell – Kappa = -0.2598922
- Waddell – MI = -0.3205067

All of the relationships between each of the co-evolution detection methods and conservation are negative, meaning that as there is an increase in conservation, there is a decrease in co-evolution score generally, as we predicted in the hypothesis 'Co-evolution score is negatively correlated with site conservation'.

Co-evolution detection method	SIFT	PolyPhen
PlotCorr	0.3201025	-0.3814595
Waddell – Kappa	0.3213145	-0.3724969
Waddell – MI	0.345617	-0.3997778

Table 8.1: Correlations between the maximum Z-scores of each of the co-evolution detection methods and the SIFT and PolyPhen scores.

SIFT/PolyPhen and co-evolution scores

As mentioned above, both SIFT and PolyPhen use conservation to calculate their scores. Consequently, there is likely to be some degree of correlation between the amount of conservation at a site, and the SIFT/PolyPhen scores at that site. The correlation between GERP_RS and the SIFT score for all of the mutations in the Fromer et al. dataset is -0.329917, and for GERP_RS and PolyPhen's HDIV score (that which uses the HumDiv dataset, henceforth just called the 'PolyPhen score'), the correlation is 0.4055978. Although both the SIFT and PolyPhen scores are reasonably correlated with the GERP conservation score, there does appear to be more going into the SIFT/PolyPhen scores than just conservation, as we would expect from the description of the scores earlier in this chapter. The correlations for both SIFT and PolyPhen are in the predicted direction; increased conservation means a mutation is predicted to be more damaging. This confirms the hypothesis 'Since SIFT and PolyPhen use conservation as one of the measures informing predicted mutation severity, there should be positive correlation between SIFT/PolyPhen mutation severity predictions and site conservation'.

Now we shall calculate the correlations between the co-evolution detection methods and SIFT and PolyPhen scores. The correlations are presented in Table 8.1.

All of the correlations indicate that mutations at sites with greater evidence for co-evolution tend to be classified as more benign by both SIFT and PolyPhen. This is as we expected in the hypothesis 'this [relationship between SIFT/PolyPhen and conservation] should be reflected in a relationship between co-evolution scores and SIFT/PolyPhen'.

Co-evolution scores of non-synonymous mutations

We now test whether (a) conservation, (b) SIFT/PolyPhen, and (c) co-evolution scores, can predict whether mutations are synonymous (silent) or non-synonymous (missense). We begin by defining a baseline model, given

by:

$$\text{mutation type} \sim \text{amino acid frequency} + \text{probability of non-synonymous} \tag{8.2}$$

against which further models are compared. The "mutation type" is a binary variable equal to 1 if the mutation is non-synonymous, and 0 if it is synonymous; "amino acid frequency" is the frequency of the amino acid at that position in all sites in our real data dataset (for human sequences); "probability of non-synonymous" is the probability that a mutation to that amino acid would be non-synonymous (would change the amino acid to something else). Other models are compared against this baseline using ANOVA, and all of the models fitted are logistic regression models, due to the fact that the variable being predicted is binary.

Running this analysis on the cases dataset only, the models we shall fit to test whether (a) conservation, (b) SIFT/PolyPhen, and (c) co-evolution scores, can predict whether mutations are synonymous or non-synonymous are:

$$\text{mutation type} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + \text{GERP\_RS} \quad \text{Conservation model} \tag{8.3}$$

$$\text{mutation type} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + \text{SIFT\_SCORE} + \text{PHEN2\_HDIV\_SCORE} \quad \text{SIFT/PolyPhen model} \tag{8.4}$$

$$\begin{aligned}\text{mutation type} \sim{}& \text{amino acid frequency} + \text{probability of non-synonymous} \\ &+ \text{plotcorr\_low} + \text{plotcorr\_z} + \text{waddellkappa\_high} + \text{waddellkappa\_low} \\ &+ \text{waddellkappa\_z} + \text{waddellmi\_high} + \text{waddellmi\_low} + \text{waddellmi\_z}\end{aligned} \quad \text{Co-evolution model} \tag{8.5}$$

respectively. GERP_RS, SIFT_SCORE and PHEN2_HDIV_SCORE are the GERP (conservation), SIFT, and PolyPhen scores, respectively. For the co-evolution model, the detection method scores ending with "_low" or "_high"

are binary variables, equal to 1 if the site was detected as co-evolving (and 0 if it was not) by that method with a low or high unique amino acid threshold, respectively. Those co-evolution detection method scores ending with "_z" are equal to the maximum Z-score calculated at that position (paired with some other position) for that detection method. PlotCorr with the higher unique amino acid threshold did not identify any sites in the cases dataset as co-evolving, which is why "plotcorr_high" does not appear in the above model.

Comparing each of the three models against the baseline model using the ANOVA (Analysis Of Variance) test, we obtain the p-values:

Model	**p-value**
Conservation	0.009306
SIFT/PolyPhen	1
Co-evolution	1

These have been corrected for 3 tests, according to the Bonferroni correction (which corrects for multiple tests). We can clearly see that the model which includes conservation is the only one of the three which is significantly different to the baseline.

As a further, exploratory analysis, we can compare a series of smaller models to the baseline model as before. We wish to do this because, as we saw previously, there is correlation between the different scores, and we want to see if any one score is having a significant effect. All of these smaller models have the format:

$$\text{mutation type} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + [\text{score}] \quad (8.6)$$

where "[score]" can be one of SIFT, PolyPhen, or one of the co-evolution scores. Again using the cases dataset, we compare each of these models to the baseline, and obtain the following p-values:

Score	p-value
SIFT_SCORE	0.3731
PHEN2_HDIV_SCORE	0.2876
plotcorr_low	0.1798
plotcorr_z	0.5749
waddellkappa_high	0.7445
waddellkappa_low	0.5611
waddellkappa_z	0.6383
waddellmi_high	0.2938
waddellmi_low	0.55
waddellmi_z	0.7372

We can see that none of the p-values is significant, and this is even before applying a Bonferroni correction. This means that none of the scores (except conservation, as we saw earlier) can predict whether mutations are synonymous or non-synonymous in the cases dataset.

Now, we investigate whether non-synonymous mutations contributing to disease occur at sites with higher (a) conservation, (b) SIFT/PolyPhen, or (c) co-evolution scores than non-synonymous mutations that do not contribute to disease. To do so, we again begin with a baseline model:

$$\text{case or control} \sim \text{amino acid frequency} + \text{probability of non-synonymous} \tag{8.7}$$

where "case or control" is a binary variable, equal to 1 if this is a case, and 0 if a control. We only consider non-synonymous mutations here. We compare the following three models to this baseline:

$$\text{case or control} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + \text{GERP\_RS} \tag{8.8}$$

$$\text{case or control} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + \text{SIFT\_SCORE} + \text{PHEN2\_HDIV\_SCORE} \tag{8.9}$$

$$\text{case or control} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + \text{plotcorr\_high} + \text{plotcorr\_low} + \text{plotcorr\_z} + \text{waddellkappa\_high} + \text{waddellkappa\_low} + \text{waddellkappa\_z} + \text{waddellmi\_high} + \text{waddellmi\_low} + \text{waddellmi\_z} \quad (8.10)$$

The only differences between these models and the models fitted previously is that the "mutation type" is replaced with "case or control". The (corrected) p-values that result from the ANOVA comparison of each of these models to the baseline are:

Model	**p-value**
Conservation	0.5418
SIFT/PolyPhen	1
Co-evolution	0.9618

None of these are statistically significant. Again, we look at the models which include each individual score, so each model has the format:

$$\text{case or control} \sim \text{amino acid frequency} + \text{probability of non-synonymous} + [\text{score}] \quad (8.11)$$

and the p-values obtained for the comparison of each of these models against the baseline are:

Score	**p-value**
SIFT_SCORE	0.5959
PHEN2_HDIV_SCORE	0.1745
plotcorr_high	0.08008
plotcorr_low	0.8114
plotcorr_z	0.2986
waddellkappa_high	0.7447
waddellkappa_low	0.6809
waddellkappa_z	0.2917
waddellmi_high	0.9538
waddellmi_low	0.7415
waddellmi_z	0.577

Even before applying the Bonferroni correction to account for multiple testing, none of the scores are able to distinguish non-synonymous mutations between the cases and controls datasets.

Comparison between diseases

We finally compare the scores of disorders (AUT = autism spectrum disorder, ID = intellectual disability, SCZ = schizophrenia) to see whether certain disorders have higher scores than others. For each of these comparisons, we use the Wilcoxon rank-sum test to first calculate whether there is a difference between the two groups (two-tailed test), and then, if there is a difference, we compare the groups again with a one-tailed test to find the direction of the difference.

The p-values for all comparisons, corrected for 9 tests (3 disease comparisons $\times$ 3 sets of scores) for the two-tailed tests are summarised in the following table:

Comparison	GERP	SIFT/PolyPhen	Co-evolution
AUT and ID	1	1	1
AUT and SCZ	1	1	0.00013374
ID and SCZ	0.29232	1	0.013113

For conservation, we compared the GERP scores between each pair of disorders. No differences were found between any of the pairs of psychiatric disorders when the p-values were corrected for multiple testing (before correction, there appears to be a difference between ID and SCZ). This is in contrast to what was found in the Fromer et al. [66] paper, where conservation scores were found to be different between disorders. This is due to the fact that this thesis only uses non-synonymous mutations; in the Fromer et al. study, loss of function and splice mutations were also used, and they almost certainly contributed to the differences found between the disorders being compared.

For SIFT/PolyPhen, we compared the union of all SIFT and PolyPhen scores for autism to all of the SIFT and PolyPhen scores for intellectual disability, etc. None of the comparisons are statistically significant (and they were not before correcting the p-values either).

Finally, we compared the Z-score co-evolution measures (plotcorr_z, waddellkappa_z, and waddellmi_z together) between pairs of disorders. We can

see differences between autism and schizophrenia, and between intellectual disability and schizophrenia. We now perform one-tailed tests to find the direction of the differences. For autism and schizophrenia, schizophrenia's Z-scores are larger than autism's (p-value = 7.431e-06), and for intellectual disability and schizophrenia, schizophrenia's Z-score's are larger than intellectual disability's (p-value = 0.0007285). This aligns with the hypothesis we set at the beginning, although we did not find differences between autism and intellectual disability.

Again, we present a secondary analysis looking at the scores individually. Comparing SIFT scores between the three diseases, it is found that mutations from schizophrenia cases have higher scores than mutations associated with intellectual disability (p-value = 0.01153). This means that mutations associated with schizophrenia would be more likely to be predicted as tolerant than mutations associated with intellectual disability. This, again, makes sense in the context of the Fromer et al. study, in that mutations that generally speaking, are associated with less severity of brain functional impairment, are tolerated more (intellectual disability tends to cause more severe impairment than schizophrenia). However, this p-value would not survive correcting for multiple tests (new p-value = 0.41508).

For PolyPhen, mutations associated with autism spectrum disorder and intellectual disability each have higher scores than those scores for mutations associated with schizophrenia (p-values 0.001075 and 0.004953 respectively). This means that mutations associated with autism spectrum disorder will be more likely to be predicted as damaging than schizophrenia, and mutations associated with intellectual disability will be more likely to be predicted as damaging than schizophrenia as well. This, again, makes sense in the context of the Fromer et al. study. Although the p-value for the intellectual disability and schizophrenia comparison would not survive correcting for multiple tests (new p-value = 0.178308), the one for autism and schizophrenia would (new p-value = 0.0387).

For the co-evolution detection methods, it is found that schizophrenia has higher scores than autism spectrum disorder for:

- binary PlotCorr with lower unique amino acid threshold (p-value = 0.008359)
- binary Waddell – Kappa with lower unique amino acid threshold (p-value = 0.005275)

- binary Waddell – MI with lower unique amino acid threshold (p-value = 0.02473)
- PlotCorr with maximum Z-scores (p-value = 0.005463)
- Waddell – Kappa with maximum Z-scores (p-value = 0.001971)
- Waddell – MI with maximum Z-scores (p-value = 0.005645)

and also schizophrenia has higher scores than intellectual disability for:

- binary PlotCorr with lower unique amino acid threshold (p-value = 0.01864)
- PlotCorr with maximum Z-scores (p-value = 0.01993)
- Waddell – MI with maximum Z-scores (p-value = 0.01731)

however, none of these would survive correction for multiple testing. Earlier in this chapter, we predicted we might expect to see the burden of less deleterious mutations in the opposite order to that seen in loss-of-function mutations, so schizophrenia, then autism spectrum disorder, then intellectual disability, with this possibly reflected in co-evolution scores. Indeed, we can see this pattern between schizophrenia and autism, and between schizophrenia and intellectual disability, but not between autism and intellectual disability. These patterns can be seen on the level of all Z-scores (from all three co-evolution detection methods) but not on the scale of any one individual method.

8.3 Summary

In this chapter we began with an overview of SIFT and PolyPhen, two alternative methods for categorising mutations as damaging/benign. We then combined the results of our proteome-wide analysis of co-evolving sites with the Fromer et al. [66] mutation dataset to produce the following findings:

1. Co-evolution score is negatively correlated with site conservation.
2. There is positive correlation between SIFT/PolyPhen mutation severity predictions and site conservation. This was also reflected in a relationship between co-evolution scores and SIFT/PolyPhen.
3. Non-synonymous mutations contributing to disease do not occur at sites with higher co-evolution scores than silent mutations, nor do they have higher scores than non-synonymous mutations not contributing to disease. The same results were found when looking at SIFT and PolyPhen scores. However, conservation scores can distinguish between non-synonymous and synonymous mutations in cases.

4. The co-evolution detection methods do tend to give different scores to mutations associated with different psychiatric disorders. In particular, schizophrenia had higher scores than autism, and schizophrenia also had higher scores than intellectual disability. This aligns with our hypothesis that 'we might expect to see the burden of less deleterious mutations in the ... order [of] schizophrenia, then autism spectrum disorder, then intellectual disability', although differences were not found between autism and intellectual disability. It should be noted that this is based on the assumption that altering protein function ("gain-of-function" mutations, potentially detected by the co-evolution score) is less deleterious (in general) to cellular function than loss-of-function. This is not necessarily the case, but the fact that a fairly robust difference is seen between schizophrenia and the other (more severe) disorders might suggest that it holds. We can see these patterns when looking at all maximum Z-scores for the three co-evolution detection methods, but not when looking at any one score individually (after correction for multiple testing).

Chapter 9

Conclusions

In this chapter we draw together the findings of this thesis in order to show how the contributions outlined at the beginning of this thesis have been achieved. We relate these contributions back to the thesis hypothesis, and consider the extent to which the thesis findings demonstrate the hypothesis. The present research has some limitations, and we conclude the chapter by identifying them and discussing how they could be addressed in future work.

9.1 A suitable method for detecting molecular co-evolution

This section relates to the contribution 'identification of a method (or a set of methods) for detecting molecular co-evolution which is suitable for use on the human proteome'.

Having reviewed the literature broadly in Chapter 3, in Chapter 6, we compared five co-evolution detection methods (CAPS, PlotCorr, MI, Waddell – Kappa, and Waddell – MI) on their performance on simulated data. (The parameters chosen for these simulations and the reasoning behind this were detailed in Chapter 5.) Simulated data was chosen for this purpose so that we could know for certain where the truly co-evolving sites are. To our knowledge, this set of methods has not been compared before, and in particular, Waddell – Kappa and Waddell – MI are previously-unimplemented methods, appearing simply as suggestions in the paper by Waddell et al. [189].

The results of this comparison were that, broadly speaking, PlotCorr and the two parametric methods (Waddell – Kappa and Waddell – MI) were the best-performing overall on the data simulated. Because the simulated data was designed to be similar to a real data set collected in Chapter 4, these

methods are also most appropriate for a proteome-wide study of molecular co-evolution.

9.2 A tree representation suitable for single-pass CUDA algorithms

This section relates to the contribution 'identification of a novel method for allowing tree-based data structures to be mapped to the CUDA architecture – this particular method is useful for when the tree only needs to be parsed (read) once'.

The two parametric methods chosen to be implemented in parallel, Waddell – Kappa and Waddell – MI, both require a binary matrix to be constructed which details the mutations which occurred on a tree describing the relationships between the input protein sequences. In order to construct this, it is necessary to inspect every branch on the tree exactly once. Although there are already algorithms for parsing a phylogenetic tree in parallel, using such a method would only make sense if we needed to parse the tree multiple times.

With this in mind, we devised a way of storing all necessary information about a tree in three arrays: one for the first children of a node, one for the second children of a node, and one for the sequence associated with that node. (We have not seen this done before.) This assumes a binary tree (one with 0, 1, or 2 children per node). Using this format of storing tree information, one can more easily access, and use, information about each branch in the tree in parallel with the others. This is further described in Chapter 7.

9.3 Parallelisation of co-evolution detection methods

This section relates to the contribution 'improved runtime performance of the chosen method(s) which allows us to look at larger data sets than has been done before, in a "reasonable" amount of time'.

In Chapter 7, a comparison was made of the serial and parallel versions of PlotCorr, Waddell – Kappa, and Waddell – MI (the parallelised versions being run on two different graphics cards, the GTX 660 Ti and the Titan X). From this comparison, it was found that often, relatively large alignments were necessary for the parallel implementations to perform better than the serial ones,

but that overall, the parallel implementations have a better cost/benefit performance. In particular, the Titan X GPU can run 6.5 times as many PlotCorr "large alignment" jobs in a month as the Raven supercomputer.

9.4 General guidelines for porting algorithms to CUDA

This section relates to the contribution 'make recommendations for how methods similar to those implemented in this thesis may be parallelised'.

9.4.1 PlotCorr

PlotCorr, as discussed previously, is a non-parametric method; that is, it does not use phylogenetic trees to inform its interpretation of the correlation between pairs of columns in an alignment. Also, its constituent steps map themselves well to functions which perform the same calculation on lots of different data items. This characteristic means that this method easily maps to CUDA, and can be re-imagined with minimal effort as a CUDA implementation.

In general, any method which calculates the set of values in a matrix for some input (and does the same set of calculations for each input) can be ported well to CUDA. This may imply that there is no good way to port parametric methods (those which use non-matrix, or tree-like, data structures) but this is not the case if we can find a way of representing the tree in another format, as described in the previous section. More generally, success in porting algorithms to CUDA will often hinge upon developing a novel insight involving transformation of a data structure into some matrix form and identifying a way in which a sequential algorithm can be modified into one or a small number of steps, each of which can be performed simultaneously on the entire data structure.

9.4.2 Parametric methods

These methods use phylogenetic trees as a source of information when calculating the correlations between pairs of sites in an alignment. For each function within a method, we must consider whether it is worth parallelising this function. When looking at a non-parametric method such as PlotCorr, it is easy to say that all of the steps will benefit from parallelisation (given a moderately large input), but with a parametric method, tree traversal is a step that cannot be as obviously mapped to CUDA. There are papers which

give methods for traversing a tree in parallel [19, 117, 77], but these assume that this is an operation we wish to do many times, whereas in our parametric methods, we only wish to do this once (and only to extract the information within the tree into a form we can more easily work with). With this in mind, we decided to keep this part of the operation in serial code form, and to parallelise the rest of the method, as these steps are more obviously parallelisable in CUDA. In particular, we devised a way (described above) of inspecting the branches of a phylogenetic tree in parallel.

9.5 Proteome-wide identification of potential co-evolving sites

This section relates to the contribution 'identification of co-evolving sites across the entire human proteome, and analysis of these sites according to what is already known about disease-causing mutations'.

In Chapter 8 we examined the results of running the co-evolution detection methods on the real data set collected in Chapter 4. We also used a data set of mutations from cases and controls from the study by Fromer et al. [66]. There were four main threads of investigation we wanted to follow; here are the results of these, as given in the summary for that chapter:

1. Co-evolution score is negatively correlated with site conservation.

2. There is positive correlation between SIFT/PolyPhen mutation severity predictions and site conservation. This was also reflected in a relationship between co-evolution scores and SIFT/PolyPhen.

3. Non-synonymous mutations contributing to disease do not occur at sites with higher co-evolution scores than silent mutations, nor do they have higher scores than non-synonymous mutations not contributing to disease. The same results were found when looking at SIFT and PolyPhen scores. However, conservation scores can distinguish between non-synonymous and synonymous mutations in cases.

4. The co-evolution detection methods do tend to give different scores to mutations associated with different psychiatric disorders. In particular, schizophrenia had higher scores than autism, and schizophrenia also had higher scores than intellectual disability. This aligns with our hypothesis that 'we might expect to see the burden of less deleterious mutations in the ... order [of] schizophrenia, then autism spectrum disorder, then intellectual disability', although differences were

not found between autism and intellectual disability. It should be noted that this is based on the assumption that altering protein function ("gain-of-function" mutations, potentially detected by the co-evolution score) is less deleterious (in general) to cellular function than loss-of-function. This is not necessarily the case, but the fact that a fairly robust difference is seen between schizophrenia and the other (more severe) disorders might suggest that it holds. We can see these patterns when looking at all maximum Z-scores for the three co-evolution detection methods, but not when looking at any one score individually (after correction for multiple testing).

9.6 Hypothesis

So far in this chapter we have linked the contributions set out in Chapter 1 to the work done in this thesis. These contributions were designed as aims to achieve to allow us to test the hypothesis set out. The hypothesis was:

'Part 1: Until this point, it has not been possible to find co-evolving sites in the entire human proteome in a feasible amount of time. We propose that translating a method for detecting co-evolving sites onto the CUDA architecture can allow us to solve this problem effectively. Part 2: Applying such a method to the human proteome shall allow us to test the hypothesis: measures of co-evolution can be used to identify functionally important sites that by their very nature cannot be found by looking at evolutionary conservation.'

The Computer Science part of the hypothesis, and the Medicine part of the hypothesis, are inherently interlinked due to the interdisciplinary nature of this work. However, it is possible to check the validity of these parts separately. For the first, that 'translating a method for detecting co-evolving sites onto the CUDA architecture can allow us to solve this problem effectively', we have proved this – in Chapter 7, we showed that implementing the PlotCorr, Waddell – Kappa, and Waddell – MI methods on the CUDA architecture allowed us to achieve an increase in performance overall. For the second part, that 'the results of applying co-evolution detection methods to the human proteome may help to identify disease-causing mutations', we showed that although the co-evolution scores could not distinguish between non-synonymous and synonymous mutations in cases, or between non-synonymous mutations in cases and controls, co-evolution detection methods are able to highlight the gradient of severity between psychiatric diseases. Schizophrenia mutations had higher co-evolution scores than autism, and also higher scores than in-

tellectual disability. Intellectual disability tends to be associated with the most severe brain functional impairment of the three, followed by autism, and then schizophrenia. The co-evolution scores show this order. Generally, the preliminary evidence presented in Chapter 8 (particularly the comparison of psychiatric disorders) suggests that the co-evolution methods can be used to identify functionally relevant sites at which mutations are more likely to be deleterious (or at least have some discernible effect).

Having shown how this thesis fulfils the contributions and tests the hypothesis set out at the beginning of the thesis, we now move on to detail the limitations of this thesis.

9.7 Limitations

In this section, we detail the limitations of this work.

To our knowledge, no co-evolution detection method more complex than Mutual Information has been implemented in CUDA previously, and we have demonstrated that it is possible to achieve substantial speed-up in a cost-effective manner by implementing these methods in CUDA. However, when implementing these parallelised methods, there came a point when we needed to stop the cycle of analysing the performance of the various functions of each method and removing bottlenecks. We have given suggestions in the next section for ways that the CUDA implementations could be further improved, but of course there may be further considerations that we could not have predicted. In addition, there is always a trade-off between the effort required to implement improvements and the improvements to efficiency that would be gained from those improvements.

When timing the various versions of PlotCorr using NSIGHT with Visual Studio, it was not possible to find an automated method of obtaining multiple timing "runs" at once. This was a limitation of developing the code in Windows, as it may be possible to perform multiple timing iterations in the Linux version of NSIGHT.

Finally, the analyses performed in Chapter 8 were preliminary; given more time we could have performed more thorough analyses of the results found; this is reserved for future work.

The limitations we have discussed here could be added to, according to the future work (anything we did not have time to do could be perceived as a limitation of this work). We list proposals for future work now.

9.8 Future work

In this section we shall detail suggestions for future work which build upon the novelty of the current work. Firstly we discuss practical considerations.

9.8.1 CUDA implementation decisions

When working out the mean and standard deviation of the correlations for our methods (whether these "correlations" are from PlotCorr or the Waddell parametric methods), we use reduction to work out the sum of the correlation values (for the mean), and of the

$$(\texttt{correlation} - \texttt{mean of correlations})^2 \tag{9.1}$$

values (for the standard deviation). There do exist more complex versions of this general method [197], but due to time constraints, these could not be pursued.

The implementations of the PlotCorr and Waddell parametric methods were not necessarily optimal, and the main purpose was to develop CUDA implementations of at least adequate efficiency (especially in the case of the parametric methods, where no actual implementations, serial or otherwise, had been done before). We have presented in this thesis considerations to bear in mind when implementing methods of this kind in CUDA, but there may be other considerations that we have not encountered here, that could be used to improve the performance of the methods further.

9.8.2 Parallelising tree-based methods further

Due to the way that the Waddell parametric methods work, although we can access information from each branch of the tree in parallel, if a node has two children, for example, two threads could attempt to access sequence data from the parent node simultaneously, causing reduced performance. In the future, it may be possible to resolve this problem.

9.8.3 PlotCorr algorithm design improvements

In the current implementation of the PlotCorr algorithm, we do not pay any attention to whether or not the alignment columns each thread is working on are conserved or not. If an alignment column is conserved, then when we get to the point of calculating the correlation between that alignment column and any other, we check to see whether the multiplied standard deviation of both of the columns is equal to 0. Fully conserved columns will always have a standard deviation equal to 0, and so we are wasting computation on calculating the distance matrix, the mean and standard deviation of this, etc., when we could simply inspect the columns and pick out only those which are not fully conserved. To reduce the number of operations we would need to complete overall, we could simply filter out only those non-conserved alignment columns, using a parallel algorithm such as compact, for example. The only difficulty which would remain is keeping some kind of mapping between the indices in the original (unfiltered) alignment, and the indices of the filtered alignment.

Another point to consider with PlotCorr is that there could be a bottleneck to this method in the form of multiple threads attempting to read the same cell of the substitution matrix, even though each thread calculates each distance matrix as a whole. This could happen if, for example, two columns in the alignment share two of the same amino acids. For example, if column 1 and column 2 both contained the amino acids D and E, then at some point the thread responsible for column 1 and the thread responsible for column 2 would each have to access the cell in the substitution matrix for amino acids D and E. (Assuming D and E appear in the two columns in the same order.) This could cause both threads to try to read from the same memory address at the same time. Improvements to stop this from happening could move in the direction of giving each thread block its own copy of the substitution matrix. Although this would not eliminate the possibility of clashes completely (there would be a clash if two threads in the same block attempted to access the same cell in the substitution matrix at the same time), it could reduce the likelihood of these clashes happening. This could also improve the overall performance as reads from shared memory (where the shared matrix would be stored) are faster than from global memory (where the substitution matrix is currently stored).

9.8.4 Timing results

In this thesis, we only considered full-program times of the serial and parallel implementations when comparing these; in the future, more time could be

spent analysing the various functions of each of the implementations and seeing how these could be improved further. In addition, a practical (as opposed to purely theoretical) investigation of alternative platforms, such as Amazon Web Services, could be performed.

Now we describe future work in terms of the biological side of this thesis.

9.8.5 Different biological applications

In this project, the focus was on intra-molecular co-evolution, i.e. co-evolution between positions within a protein. A future application could be to use our CUDA programs to look at inter-molecular co-evolution (co-evolution between positions in different proteins).

9.8.6 Extensions

One way we could further analyse the results on the real data set is that, given the set of Z-scores associated with pairs of positions in an alignment, we could cluster these according to high/low correlation to work out networks of co-evolving sites. This could elucidate more information on how different parts of a protein work together.

9.8.7 Different datasets

A different real data dataset, such as one from the UCSC website, could have been chosen; this would give us different results and perhaps differing conclusions could be drawn.

9.9 Summary

In this thesis we have shown that it is possible to use CUDA to perform cost-effective, large-scale analyses of molecular sequences in order to identify potentially co-evolving sites. These techniques have the potential to make an impact in the understanding of psychiatric disorders such as schizophrenia, autism, and intellectual disability, and potential treatments.

Chapter 10

Bibliography

[1] Sharon H. Ackerman, Elisabeth R. Tillier, and Domenico L. Gatti. Accurate simulation and detection of coevolution signals in multiple sequence alignments. *PloS one*, 7(10):e47108, 2012.

[2] Jun Adachi and Masami Hasegawa. Model of Amino Acid Substitution in Proteins Encoded by Mitochondrial DNA. *Journal of molecular evolution*, 42(4):459–468, 1996.

[3] Ivan Adzhubei, Daniel M. Jordan, and Shamil R. Sunyaev. Predicting Functional Effect of Human Missense Mutations Using PolyPhen-2. *Current Protocols in Human Genetics*, (SUPPL.76), 2013.

[4] Ivan A. Adzhubei, Steffen Schmidt, Leonid Peshkin, Vasily E. Ramensky, Anna Gerasimova, Peer Bork, Alexey S. Kondrashov, and Shamil R. Sunyaev. A method and server for predicting damaging missense mutations. *Nature methods*, 7(4):248–249, 2010.

[5] Robin G. Allaby and Mathew Woodwark. Phylogenomic Analysis Reveals Extensive Phylogenetic Mosaicism in the Human GPCR Superfamily. *Evolutionary Bioinformatics*, 3:357–370, 2007.

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

[7] 'Apers0n'. DNA Double Helix. `https://commons.wikimedia.org/wiki/File:DNA_Double_Helix.png`. Derivative work. Original work released by National Human Genome Research Institute at URL `https://www.genome.gov/dmd/img.cfm?node=Photos/Graphics&id=85329`. This is in the Public Domain. Accessed on 19/08/16.

[8] Miguel Arenas, Helena G. Dos Santos, David Posada, and Ugo Bastolla. Protein evolution along phylogenetic histories under structurally constrained substitution models. *Bioinformatics (Oxford, England)*, 29(23):3020–8, 2013.

[9] William R. Atchley, Werner Terhalle, and Andreas Dress. Positional Dependence, Cliques, and Predictive Motifs in the bHLH Protein Domain. *Journal of Molecular Evolution*, 48(5):501–516, 1999.

[10] William R. Atchley, Kurt R. Wollenberg, Walter M. Fitch, Werner Terhalle, and Andreas W. Dress. Correlations Among Amino Acid Sites in bHLH Protein Domains: An Information Theoretic Analysis. *Mol Biol Evol*, 17(1):164–178, 2000.

[11] 'Autopilot'. Probability density plots of gamma distributions. `https://en.wikipedia.org/wiki/Gamma_distribution#/media/File:Gamma_distribution_pdf.svg`. Derivative work. Original work by 'MarkSweep and 'Cburnett'. Accessed on 08/08/16. Licensed under the Creative Commons Attribution-Share Alike 3.0 Unported licence, found at https://creativecommons.org/licenses/by-sa/3.0/legalcode (human-readable version at https://creativecommons.org/licenses/by-sa/3.0/deed.en).

[12] M. Madan Babu. The contribution of intrinsically disordered regions to protein function, cellular complexity, and human disease. *Biochemical Society Transactions*, 44(5):1185–1200, 2016.

[13] 'Bilou'. blue Alpha Helix Section. `https://commons.wikimedia.org/wiki/File:AlphaHelixSection_(blue).svg`. Accessed on 04/11/16. This is in the Public Domain.

[14] Thomas Bradley. Assess, Parallelize, Optimize, Deploy. `https://devblogs.nvidia.com/parallelforall/assess-parallelize-optimize-deploy/`. [Online; accessed 2016-10-12].

[15] Catherine Brooksbank, Evelyn Camon, Midori A. Harris, Michele Magrane, Maria Jesus Martin, Nicola Mulder, Claire O'Donovan, Helen Parkinson, Mary Ann Tuli, Rolf Apweiler, Ewan Birney, Alvis Brazma, Kim Henrick, Rodrigo Lopez, Guenter Stoesser, Peter Stoehr, and Graham Cameron. The European Bioinformatics Institute's data resources. *Nucleic Acids Research*, 31(1):43–50, 2003.

[16] Christopher A. Brown and Kevin S. Brown. Validation of coevolving residue algorithms via pipeline sensitivity analysis: ELSC and OMES and ZNMI, oh my! *PloS one*, 5(6):e10779, 2010.

[17] Michael J. Buck and William R. Atchley. Networks of Coevolving Sites in Structural and Functional Domains of Serpin Proteins. *Molecular Biology and Evolution*, 22(7):1627–1634, 2005.

[18] Lukas Burger and Erik van Nimwegen. Disentangling Direct from Indirect Co-Evolution of Residues in Protein Alignments. *PLoS Computational Biology*, 6(1), 2010.

[19] Martin Burtscher and Keshav Pingali. An Efficient CUDA Implementation of the Tree-Based Barnes Hut n-Body Algorithm. In Wen-Mei W. Hwu, editor, *GPU Computing Gems Emerald Edition*, pages 75–92. Morgan Kaufmann, 2011.

[20] Cristina Marino Buslje, Javier Santos, Jose Maria Delfino, and Morten Nielsen. Correction for phylogeny, small number of observations and data redundancy improves the identification of coevolving amino acid pairs using mutual information. *Bioinformatics (Oxford, England)*, 25(9):1125–31, 2009.

[21] Ewen Callaway. 'Junk' DNA gets credit for making us who we are. `https://www.newscientist.com/article/dn18680-junk-dna-gets-credit-for-making-us-who-we-are/?full=true{&}print=true`. [Online; accessed 2016-08-16].

[22] J. Gregory Caporaso, Sandra Smit, Brett C. Easton, Lawrence Hunter, Gavin A. Huttley, and Rob Knight. Detecting coevolution without phylogenetic trees? Tree-ignorant metrics of coevolution perform as well as tree-aware metrics. *BMC Evolutionary Biology*, 8:327, 2008.

[23] Center for Comparative Genomics and Bioinformatics (Pennsylvania State University). BLASTZ is Obsolete. `http://www.bx.psu.edu/miller{_}lab/dist/blastz{_}is{_}obsolete.html`. [Online; accessed 2016-08-17].

[24] Saikat Chakrabarti and Anna R. Panchenko. Coevolution in defining the functional specificity. *Proteins*, 75(1):231–40, 2009.

[25] Saikat Chakrabarti and Anna R. Panchenko. Structural and functional roles of coevolved sites in proteins. *PloS one*, 5(1):e8591, 2010.

[26] Jia-Ming Chang, Paolo Di Tommaso, Vincent Lefort, Olivier Gascuel, and Cedric Notredame. TCS: a web server for multiple sequence alignment evaluation and phylogenetic reconstruction. *Nucleic acids research*, 43(W1):W3–6, 2015.

[27] Jia-Ming Chang, Paolo Di Tommaso, and Cedric Notredame. TCS: A New Multiple Sequence Alignment Reliability Measure to Estimate Alignment Accuracy and Improve Phylogenetic Tree Reconstruction. *Molecular Biology and Evolution*, 31(6):1625–1637, 2014.

[28] Gareth Chelvanayagam, Andreas Eggenschwiler, Lukas Knecht, Gaston H. Gonnet, and Steven A. Benner. An analysis of simultaneous variation in protein structures. *Protein engineering*, 10(4):307–16, 1997.

[29] Sun Shim Choi, Weimin Li, and Bruce T Lahn. Robust signals of coevolution of interacting residues in mammalian proteomes identified by phylogeny-aided structural analysis. *Nature genetics*, 37(12):1367–71, 2005.

[30] Greg W. Clark, Sharon H. Ackerman, Elisabeth R. Tillier, and Domenico L. Gatti. Multidimensional mutual information methods for the analysis of covariation in multiple sequence alignments. *BMC Bioinformatics*, 15(1):157, 2014.

[31] Francisco M. Codoñer and Mario A. Fares. Why Should We Care About Molecular Coevolution? *Evolutionary Bioinformatics Online*, 4:29–38, 2008.

[32] Francisco M. Codoñer, Mario A. Fares, and Elena F. Santiago. Adaptive Covariation between the Coat and Movement Proteins of Prunus Necrotic Ringspot Virus. *Journal of virology*, 80(12):5833–40, 2006.

[33] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[34] Gregory M. Cooper, Eric A. Stone, George Asimenos, Eric D. Green, Serafim Batzoglou, and Arend Sidow. Distribution and intensity of constraint in mammalian genomic sequence. *Genome Research*, 15(7):901–913, 2005.

[35] Daniel A. Dalquen, Maria Anisimova, Gaston H. Gonnet, and Christophe Dessimoz. ALF–A Simulation Framework for Genome Evolution. *Molecular biology and evolution*, 29(4):1115–23, 2012.

[36] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. 22 A Model of Evolutionary Change in Proteins. In *Atlas of Protein Sequence and Structure*, pages 345–352. 1978.

[37] David de Juan, Florencio Pazos, and Alfonso Valencia. Emerging methods in protein co-evolution. *Nature reviews. Genetics*, 14(4):249–61, 2013.

[38] John P. Dekker, Anthony Fodor, Richard W. Aldrich, and Gary Yellen. A perturbation-based method for calculating explicit likelihood of evolutionary co-variance in multiple sequence alignments. *Bioinformatics*, 20(10):1565–1572, 2004.

[39] Die.net. clock_gettime(3) - Linux max page. `https://linux.die.net/man/3/clock{_}gettime`.

[40] Matthew W. Dimmic, Melissa J. Hubisz, Carlos D. Bustamante, and Rasmus Nielsen. Detecting coevolving amino acid sites using Bayesian mutational mapping. *Bioinformatics (Oxford, England)*, 21(Suppl 1):i126–i135, 2005.

[41] Emmanuel J. P. Douzery, Elizabeth A. Snell, Eric Bapteste, Frédéric Delsuc, and Hervé Philippe. The timing of eukaryotic evolution: Does a relaxed molecular clock reconcile proteins and fossils? *Proceedings of the National Academy of Sciences of the United States of America*, 101(43):15386–15391, 2004.

[42] S. D. Dunn, L. M. Wahl, and G. B. Gloor. Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. *Bioinformatics (Oxford, England)*, 24(3):333–40, 2008.

[43] Julien Dutheil and Nicolas Galtier. Detecting groups of coevolving positions in a molecule: a clustering approach. *BMC evolutionary biology*, 7:242, 2007.

[44] Julien Dutheil, Tal Pupko, Alain Jean-Marie, and Nicolas Galtier. A model-based approach for detecting coevolving positions in a molecule. *Molecular biology and evolution*, 22(9):1919–28, 2005.

[45] Julien Y. Dutheil. Detecting coevolving positions in a molecule: why and how to account for phylogeny. *Briefings in Bioinformatics*, 13(2):228–243, 2012.

[46] B. C. Easton, A. V. Isaev, G. A. Huttley, and P. Maxwell. A probabilistic method to identify compensatory substitutions for pathogenic mutations. In David Sankoff, Lusheng Wang, and Francis Chin, editors, *Proceedings of the 5th Asia-Pacific Bioinformatics Conference, volume 5 of Advances in Bioinformatics and Computational Biology*, pages 195–205, Hong Kong, 2007. Imperial College Press.

[47] Sean R. Eddy. Where did the BLOSUM62 alignment score matrix come from? *Nature biotechnology*, 22(8):1035–6, 2004.

[48] R. J. Edwards and J. D. Parker. UPGMA Worked Example. `http://www.southampton.ac.uk/{~}re1u06/teaching/upgma/`. [Online; accessed 2016-10-14].

[49] Paul R. Ehrlich and Peter H. Raven. Butterflies and Plants: A Study in Coevolution. *Evolution*, 18(4):586–608, 1964.

[50] EMBL-EBI. EMBL-EBI:: Help:: Tools:: ClustalW2 Phylogeny FAQ. `http://www.ebi.ac.uk/Tools/phylogeny/clustalw2{_}phylogeny/help/faq.html`. [Online; accessed 2016-07-13].

[51] EMBL-EBI. Help & Documentation. `http://www.ensembl.org/info/index.html`. [Online; accessed 2017-06-05].

[52] EMBL-EBI. Species List. `http://www.ensembl.org/info/about/species.html`. [Online; accessed 2017-06-05].

[53] EMBL-EBI. UniProtKB/TrEMBL Release Notes: Release 24, June 2003. `http://www.ebi.ac.uk/uniprot/TrEMBLdocs/trembl{_}release{_}notes{_}24.html`. [Online; accessed 2017-04-25].

[54] Ensembl. Protein trees and orthologies. `http://www.ensembl.org/info/genome/compara/homology{_}method.html`. [Online; accessed 2014-07-09].

[55] Chiara Fabbri and Alessandro Serretti. Role of 108 schizophrenia-associated loci in modulating psychopathological dimensions in schizophrenia and bipolar disorder. *American Journal of Medical Genetics Part B: Neuropsychiatric Genetics*, 174(7):757–764, 2017.

[56] Mario A. Fares and Simon A. A. Travers. A Novel Method for Detecting Intramolecular Coevolution: Adding a Further Dimension to Selective Constraints Analyses. *Genetics*, 173(1):9–23, 2006.

[57] Joseph Felsenstein. Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *Journal of Molecular Evolution*, 17(6):368–376, 1981.

[58] Andrew D. Fernandes and Gregory B. Gloor. Mutual information is critically dependent on prior assumptions: would the correct estimate of mutual information please identify itself? *Bioinformatics (Oxford, England)*, 26(9):1135–9, 2010.

[59] Walter M. Fitch. Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology. *Systematic Biology*, 20(4):406–416, 1971.

[60] Sarel J. Fleishman, Vinzenz M. Unger, Mark Yeager, and Nir Ben-Tal. A Calpha Model for the Transmembrane alpha Helices of Gap Junction Intercellular Channels. *Molecular Cell*, 15(6):879–888, 2004.

[61] Sarel J. Fleishman, Ofer Yifrach, and Nir Ben-Tal. An Evolutionarily Conserved Network of Amino Acids Mediates Gating in Voltage-dependent Potassium Channels. *Journal of Molecular Biology*, 340(2):307–318, 2004.

[62] William Fletcher and Ziheng Yang. INDELible: A Flexible Simulator of Biological Sequence Evolution. *Molecular biology and evolution*, 26(8):1879–88, 2009.

[63] Paul Flicek, M. Ridwan Amode, Daniel Barrell, Kathryn Beal, Konstantinos Billis, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah Hunt, Nathan Johnson, Thomas Juettemann, Andreas K. Kähäri, Stephen Keenan, Eugene Kulesha, Fergal J. Martin, Thomas Maurel, William M. McLaren, Daniel N. Murphy, Rishi Nag, Bert Overduin, Miguel Pignatelli, Bethan Pritchard, Emily Pritchard, Harpreet S. Riat, Magali Ruffier, Daniel Sheppard, Kieron Taylor, Anja Thormann, Stephen J. Trevanion, Alessandro Vullo, Steven P. Wilder, Mark Wilson, Amonida Zadissa, Bronwen L. Aken, Ewan Birney, Fiona Cunningham, Jennifer Harrow, Javier Herrero, Tim J P Hubbard, Rhoda Kinsella, Matthieu Muffato, Anne Parker, Giulietta Spudich, Andy Yates, Daniel R. Zerbino, and Stephen M. J. Searle. Ensembl 2014. *Nucleic Acids Research*, 42(D1):749–755, 2014.

[64] Michael J. Flynn. Some Computer Organizations and Their Effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, 1972.

[65] Anthony A. Fodor and Richard W. Aldrich. Influence of Conservation on Calculations of Amino Acid Covariance in Multiple Sequence Alignments. *Proteins*, 56(2):211–21, 2004.

[66] Menachem Fromer, Andrew J. Pocklington, David H. Kavanagh, Hywel J. Williams, Sarah Dwyer, Padhraig Gormley, Lyudmila Georgieva, Elliott Rees, Priit Palta, Douglas M. Ruderfer, Noa Carrera, Isla Humphreys, Jessica S. Johnson, Panos Roussos, Douglas D. Barker, Eric Banks, Vihra Milanova, Seth G. Grant, Eilis Hannon, Samuel A. Rose, Kimberly Chambert, Milind Mahajan, Edward M. Scolnick, Jennifer L. Moran, George Kirov, Aarno Palotie, Steven A. McCarroll, Peter Holmans, Pamela Sklar, Michael J. Owen, Shaun M. Purcell, and Michael C. O'Donovan. De novo mutations in schizophrenia implicate synaptic networks. *Nature*, 506(7487):179–184, 2014.

[67] Angelika Fuchs, Antonio J. Martin-Galiano, Matan Kalman, Sarel Fleishman, Nir Ben-Tal, and Dmitrij Frishman. Co-evolving residues in membrane proteins. *Bioinformatics*, 23(24):3312–3319, 2007.

[68] K. Fukami-Kobayashi, D. R. Schreiber, and S. A. Benner. Detecting Compensatory Covariation Signals in Protein Evolution Using Reconstructed Ancestral Sequences. *Journal of Molecular Biology*, 319(3):729–743, 2002.

[69] Hongyun Gao, Yongchao Dou, Jialiang Yang, and Jun Wang. New methods to measure residues coevolution in proteins. *BMC Bioinformatics*, 12(1):206, 2011.

[70] Giulio Genovese, Menachem Fromer, Eli A Stahl, Douglas M Ruderfer, Kimberly Chambert, Mikael Landén, Jennifer L Moran, Shaun M Purcell, Pamela Sklar, Patrick F Sullivan, Christina M Hultman, and Steven A McCarroll. Increased burden of ultra-rare protein-altering variants among 4,877 individuals with schizophrenia. *Nature Neuroscience*, 19(11):1433–1441, 2016.

[71] Asghar Ghasemi and Saleh Zahediasl. Normality Tests for Statistical Analysis: A Guide for Non-Statisticians. *International Journal of Endocrinology and Metabolism*, 10(2):486–489, 2012.

[72] D. G. Gilbert. Phylodendron - Phylogenetic tree printer. `http://iubio.bio.indiana.edu/treeapp/treeprint-form.html`.

[73] Gregory B. Gloor, Louise C. Martin, Lindi M. Wahl, and Stanley D. Dunn. Mutual Information in Protein Multiple Sequence Alignments Reveals Two Classes of Coevolving Positions. *Biochemistry*, 44(19):7156–65, 2005.

[74] Gregory B. Gloor, Gaurav Tyagi, Dana M. Abrassart, Andrew J. Kingston, Andrew D. Fernandes, Stanley D. Dunn, and Christopher J. Brandl. Functionally compensating coevolving positions are neither homoplasic nor conserved in clades. *Molecular biology and evolution*, 27(5):1181–91, 2010.

[75] Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated Mutations and Residue Contacts in Proteins. *Proteins: Structure, Function, and Genetics*, 18(4):309–317, 1994.

[76] Chern-Sing Goh, Andrew A. Bogan, Marcin Joachimiak, Dirk Walther, and Fred E. Cohen. Co-evolution of proteins with their interaction partners. *Journal of molecular biology*, 299(2):283–93, 2000.

[77] Michael Goldfarb, Youngjoon Jo, and Milind Kulkarni. General Transformations for GPU Execution of Tree Traversals. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*, pages 1–12, 2013.

[78] Rodrigo Gouveia-Oliveira and Anders G. Pedersen. Finding coevolving amino acid residues using row and column weighting of mutual information and multi-dimensional amino acid representation. *Algorithms for molecular biology : AMB*, 2:12, 2007.

[79] Johan A. Grahnen and David A. Liberles. CASS: Protein sequence simulation with explicit genotype-phenotype mapping. *Trends in Evolutionary Biology*, 4(1):47–49, 2012.

[80] Nicholas C. Grassly, Jun Adachi, and Andrew Rambaut. PSeq-Gen: an application for the Monte Carlo simulation of protein sequence evolution along phylogenetic trees. *Computer applications in the biosciences*, 13(5):559–560, 1997.

[81] Nicholas C. Grassly, Jun Adachi, Andrew Rambaut, and P. Tuffery. CS-PSeq-Gen. `http://bioserv.rpbs.univ-paris-diderot.fr/software/CS-PSeq-Gen/`, 2002.

[82] Thasso Griebel, Benedikt Zacher, Paolo Ribeca, Emanuele Raineri, Vincent Lacroix, Roderic Guigó, and Michael Sammeth. Modelling

and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic acids research*, 40(20):10073–83, 2012.

[83] Attila Gulyás-Kovács. Integrated Analysis of Residue Coevolution and Protein Structure in ABC Transporters. *PLoS ONE*, 7(5):e36546, 2012.

[84] Najeeb Halabi, Olivier Rivoire, Stanislas Leibler, and Rama Ranganathan. Protein Sectors: Evolutionary Units of Three-Dimensional Structure. *Cell*, 138(4):774–786, 2009.

[85] Barry G. Hall. Simulating DNA Coding Sequence Evolution with EvolveAGene 3. *Molecular biology and evolution*, 25(4):688–95, 2008.

[86] Nicholas Hamilton, Kevin Burrage, Mark A. Ragan, and Thomas Huber. Protein Contact Prediction Using Patterns of Correlation. *Proteins: Structure, Function, and Bioinformatics*, 56(4):679–684, 2004.

[87] Dehua Hang, Eric Torng, Charles Ofria, and Thomas M. Schmidt. The effect of natural selection on the performance of maximum parsimony. *BMC Evolutionary Biology*, 7:94, 2007.

[88] Mark Jason Harris. *Real-Time Cloud Simulation and Rendering*. PhD thesis, University of North Carolina, 2003.

[89] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–9, 1992.

[90] Fred' 'Hsu. The Ancestors Tale Mammals cladogram. `https://commons.wikimedia.org/wiki/File:The_Ancestors_Tale_Mammals_cladogram.png`. Accessed on 06/04/17. Licensed under the GNU Free Documentation License, found at https://www.gnu.org/licenses/fdl.html.

[91] Jing Hu and Pauline C. Ng. Predicting the effects of frameshifting indels. *Genome Biology*, 13(2):R9, 2012.

[92] Jing Hu and Pauline C. Ng. SIFT Indel: Predictions for the Functional Effects of Amino Acid Insertions/Deletions in Proteins. *PloS one*, 8(10):e77940, 2013.

[93] Yuji Inagaki, Christian Blouin, Edward Susko, and Andrew J. Roger. Assessing functional divergence in EF-1α and its paralogs in eukaryotes and archaebacteria. *Nucleic Acids Research*, 31(14):4227–4237, 2003.

[94] UCLA Institute for Digital Research and Education. What statistical analysis should I use? `http://www.ats.ucla.edu/stat/mult{_}pkg/whatstat/`. [Online; accessed 2017-04-04].

[95] Chan-Seok Jeong and Dongsup Kim. Reliable and robust detection of coevolving protein residues. *Protein engineering, design & selection : PEDS*, 25(11):705–13, 2012.

[96] David T. Jones, Daniel W. A. Buchan, Domenico Cozzetto, and Massimiliano Pontil. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[97] David T. Jones, William R. Taylor, and Janet M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Computer Applications in the Biosciences*, 8(3):275–282, 1992.

[98] Jeffrey B. Joy, Richard H. Liang, Rosemary M. McCloskey, T. Nguyen, and Art F. Y. Poon. Ancestral Reconstruction. *PLoS Computational Biology*, 12(7):1–20, 2016.

[99] Itamar Kass and Amnon Horovitz. Mapping Pathways of Allosteric Communication in GroEL by Analysis of Correlated Mutations. *Proteins: Structure, Function and Genetics*, 48(4):611–617, 2002.

[100] W. James Kent, Robert Baertsch, Angie Hinrichs, Webb Miller, and David Haussler. Evolution's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes. *Proceedings of the National Academy of Sciences of the United States of America*, 100(20):11484–11489, 2003.

[101] Wan K. Kim, Dan M. Bolser, and Jong H. Park. Large-scale coevolution analysis of protein structural interlogues using the global protein structural interactome map (PSIMAP). *Bioinformatics (Oxford, England)*, 20(7):1138–1150, 2004.

[102] G Kirov, A J Pocklington, P Holmans, D Ivanov, M Ikeda, D Ruderfer, J Moran, K Chambert, D Toncheva, L Georgieva, D Grozeva, M Fjodorova, R Wollerton, E Rees, I Nikolov, L N van de Lagemaat, À Bayés, E Fernandez, P I Olason, Y Böttcher, N H Komiyama, M O Collins, J Choudhary, K Stefansson, H Stefansson, S G N Grant, S Purcell, P Sklar, M C O'Donovan, and M J Owen. De novo CNV analysis implicates specific abnormalities of postsynaptic signalling complexes in the pathogenesis of schizophrenia. *Molecular Psychiatry*, 17(2):142–153, 2012.

[103] Bette T. M. Korber, Robert M. Farber, David H. Wolpert, and Alan S. Lapedes. Covariation of mutations in the V3 loop of human immunodeficiency virus type 1 envelope protein: An information theoretic analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 90(15):7176–7180, 1993.

[104] Andreas Kowarsch, Angelika Fuchs, Dmitrij Frishman, and Philipp Pagel. Correlated mutations: a hallmark of phenotypic amino acid substitutions. *PLoS computational biology*, 6(9):e1000923, 2010.

[105] Prateek Kumar, Steven Henikoff, and Pauline C. Ng. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nature Protocols*, 4(7):1073–1081, 2009.

[106] Laerd Statistics. Wilcoxon Signed-Rank Test using SPSS Statistics. `https://statistics.laerd.com/spss-tutorials/wilcoxon-signed-rank-test-using-spss-statistics.php`.

[107] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. Mcgettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, and D. G. Higgins. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[108] Michael S. Y. Lee. Unalignable sequences and molecular evolution. *Trends in Ecology and Evolution*, 16(12):681–685, 2001.

[109] Christopher Lewis. Definition of Homolog, Ortholog and Paralog. `http://homepage.usask.ca/{~}ctl271/857/def{_}homolog.shtml`.

[110] Juntao Liu, Xiaoyun Duan, Jianyang Sun, Yanbin Yin, Guojun Li, Lushan Wang, and Bingqiang Liu. Bi-Factor Analysis Based on Noise-Reduction (BIFANR): A New Algorithm for Detecting Coevolving Amino Acid Sites in Proteins. *PloS one*, 8(11):e79764, 2013.

[111] Steve W. Lockless and Rama Ranganathan. Evolutionarily Conserved Pathways of Energetic Connectivity in Protein Families. *Science*, 286(5438):295–299, 1999.

[112] Ari Löytynoja. Alignment Methods: Strategies, Challenges, Benchmarking, and Comparative Overview. In Maria Anisimova, editor, *Evolutionary Genomics*, volume 855, pages 203–235. Humana Press, New York, 1 edition, 2012.

[113] Bob MacCallum. Sequence conservation. http://www.sbc.su.se/{~}maccallr/thesis/node112.html. [Online; accessed 2017-01-05].

[114] Wenzhi Mao, Cihan Kaya, Anindita Dutta, Amnon Horovitz, and Ivet Bahar. Comparative study of the effectiveness and limitations of current methods for detecting sequence coevolution. *Bioinformatics*, 31(12):1929–1937, 2015.

[115] Debora S. Marks, Lucy J. Colwell, Robert Sheridan, Thomas A. Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3D structure computed from evolutionary sequence variation. *PLoS ONE*, 6(12), 2011.

[116] L. C. Martin, G. B. Gloor, S. D. Dunn, and L. M. Wahl. Using information theory to search for co-evolving residues in proteins. *Bioinformatics (Oxford, England)*, 21(22):4116–24, 2005.

[117] Wellington S. Martins, Thiago F. Rangel, Divino C. S. Lucas, Elias B. Ferreira, and Edson N. Caceres. Phylogenetic Distance Computation Using CUDA. In *Advances in Bioinformatics and Computational Biology*, pages 168–178. 2012.

[118] Timothy Mattson. How to sound like a Parallel Programming Expert Part 2: Parallel hardware. https://software.intel.com/en-us/articles/how-to-sound-like-a-parallel-programming-expert-part-2-parallel-hardware. [Online; accessed 2017-02-27].

[119] Michael McCool, Arch Robison, and James Reinders. Amdahl's Law vs. Gustafson-Barsis' Law. http://www.drdobbs.com/parallel/amdahls-law-vs-gustafson-barsis-law/240162980. [Online; accessed 2016-10-12].

[120] A.D. McLachlan. Tests for comparing related amino-acid sequences. Cytochrome c and cytochrome c551. *Journal of molecular biology*, 61(2):409–424, 1971.

[121] Rainer Merkl and Matthias Zwick. H2r: Identification of evolutionary important residues by means of an entropy based analysis of multiple sequence alignments. *BMC Bioinformatics*, 9:151, 2008.

[122] Merriam-Webster. synapse. https://www.merriam-webster.com/dictionary/synapse. [Online; accessed 2017-09-25].

[123] Lisa A. Miosge, Matthew A. Field, Yovina Sontani, Vicky Cho, Simon Johnson, Anna Palkova, Bhavani Balakishnan, Rong Liang, Yafei Zhang, Stephen Lyon, Bruce Beutler, Belinda Whittle, Edward M. Bertram, Anselm Enders, Christopher C. Goodnow, and T. Daniel Andrews. Comparison of predicted and actual consequences of missense mutations. *Proceedings of the National Academy of Sciences of the United States of America*, 112(37):E5189–98, 2015.

[124] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S. Marks, Chris Sander, Riccardo Zecchina, José N. Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences of the United States of America*, 108(49):E1293–301, 2011.

[125] David A. Morrison. How to Summarize Estimates of Ancestral Divergence Times. *Evolutionary Bioinformatics*, 4:75–95, 2008.

[126] National Institutes of Health. Genetic Simulation Resources Home. `https://popmodels.cancercontrol.cancer.gov/gsr/`.

[127] Nature Education. species. `http://www.nature.com/scitable/definition/species-312`. [Online; accessed 2016-11-07].

[128] Nature Education. synteny — Learn Science at Scitable. `http://www.nature.com/scitable/definition/synteny-137`. [Online; accessed 2014-07-09].

[129] NCBI. Build procedure -. `http://www.ncbi.nlm.nih.gov/homologene/build-procedure/`.

[130] NCBI. Home - HomoloGene - NCBI. `http://www.ncbi.nlm.nih.gov/homologene`.

[131] Erwin Neher. How frequent are correlated changes in families of protein sequences? *Proceedings of the National Academy of Sciences of the United States of America*, 91(1):98–102, 1994.

[132] Pauline C. Ng and Steven Henikoff. Predicting deleterious amino acid substitutions. *Genome research*, 11(5):863–74, 2001.

[133] Pauline C. Ng and Steven Henikoff. Predicting the Effects of Amino Acid Substitutions on Protein Function. *Annual Review of Genomics and Human Genetics*, 7:61–80, 2006.

[134] R Nielsen and Z Yang. Likelihood models for detecting positively selected amino acid sites and applications to the HIV-1 envelope gene. *Genetics*, 148(3):929–36, 1998.

[135] Orly Noivirt, Miriam Eisenstein, and Amnon Horovitz. Detection and reduction of evolutionary noise in correlated mutation analysis. *Protein Engineering, Design and Selection*, 18(5):247–253, 2005.

[136] Cédric Notredame, Desmond G. Higgins, and Jaap Heringa. T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. *Journal of Molecular Biology*, 302(1):205–217, 2000.

[137] NVIDIA. NVIDIA TITAN X Graphics Card with Pascal — GeForce. `http://www.geforce.co.uk/hardware/10series/titan-x/`. [Online; accessed 2016-10-14].

[138] David Ochoa, David Juan, Alfonso Valencia, and Florencio Pazos. Detection of significant protein co-evolution. *Bioinformatics (Oxford, England)*, pages 1–8, 2015.

[139] Michael Conlon O'Donovan and Michael John Owen. The implications of the shared genetics of psychiatric disorders. *Nature Medicine*, 22(11):1214–1219, 2016.

[140] Osvaldo Olmea, Burkhard Rost, and Alfonso Valencia. Effective Use of Sequence Correlation and Conservation in Fold Recognition. *Journal of molecular biology*, 293(5):1221–39, 1999.

[141] Osvaldo Olmea and Alfonso Valencia. Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Folding & Design*, 2(3):S25–S32, 1997.

[142] Michael J. Owen, Michael C. O'Donovan, Anita Thapar, and Nicholas Craddock. Neurodevelopmental hypothesis of schizophrenia. *The British Journal of Psychiatry*, 198(3):173–175, 2011.

[143] Mark Pagel. Detecting Correlated Evolution on Phylogenies: A General Method for the Comparative Analysis of Discrete Characters. *Proceedings: Biological Sciences*, 255(1342):37–45, 1994.

[144] Fabiano Sviatopolk-Mirsky Pais, Patrícia De Cássia Ruy, Guilherme Oliveira, and Roney Santos Coimbra. Assessing the efficiency of multiple sequence alignment programs. *Algorithms for Molecular Biology*, 9(4), 2014.

[145] Andy Pang, Andrew D. Smith, Paulo A. S. Nuin, and Elisabeth R. M. Tillier. SIMPROT: Using an empirically determined indel distribution in simulations of protein evolution. *BMC bioinformatics*, 6:236, 2005.

[146] Florencio Pazos and Alfonso Valencia. Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein engineering*, 14(9):609–614, 2001.

[147] Florencio Pazos and Alfonso Valencia. Protein co-evolution, co-adaptation and interactions. *The EMBO journal*, 27(20):2648–55, 2008.

[148] PCUpgrade. Gigabyte Nvidia GTX 660 Ti Windforce2X 2GB GDDR5 Dual Link DVI HDMI Display Port PCI-E Retail Graphics Card [accessed through the Internet Archive]. `http://web.archive.org/web/20160804092219/http://www.pcupgrade.co.uk/productdetails.asp?productid=11186{&}categoryid=538`. [Online; accessed 2016-08-04].

[149] PCUpgrade. MSI Nvidia GTX 660 Ti Power Edition 2GB GDDR5 Dual DVI HDMI DisplayPort PCI-E Retail Graphics Card [accessed through the Internet Archive]. `http://web.archive.org/web/20160804092014/http://www.pcupgrade.co.uk/productdetails.asp?productid=11187{&}categoryid=538`. [Online; accessed 2016-08-04].

[150] PCUpgrade. Zotac Nvidia GTX 660 Ti 2048MB GDDR5 Dual link DVI HDMI DisplayPort PCI-E Retail Graphics Card [accessed through the Internet Archive]. `http://web.archive.org/web/20160804092140/http://www.pcupgrade.co.uk/productdetails.asp?productid=11189{&}categoryid=538`. [Online; accessed 2016-08-04].

[151] Dalila Pinto, Elsa Delaby, Daniele Merico, Mafalda Barbosa, Alison Merikangas, Lambertus Klei, Bhooma Thiruvahindrapuram, Xiao Xu, Robert Ziman, Zhuozhi Wang, Jacob A.S. Vorstman, Ann Thompson, Regina Regan, Marion Pilorge, Giovanna Pellecchia, Alistair T. Pagnamenta, Bárbara Oliveira, Christian R. Marshall, Tiago R. Magalhaes, Jennifer K. Lowe, Jennifer L. Howe, Anthony J. Griswold, John Gilbert, Eftichia Duketis, Beth A. Dombroski, Maretha V. De

Jonge, Michael Cuccaro, Emily L. Crawford, Catarina T. Correia, Judith Conroy, Inês C. Conceição, Andreas G. Chiocchetti, Jillian P. Casey, Guiqing Cai, Christelle Cabrol, Nadia Bolshakova, Elena Bacchelli, Richard Anney, Steven Gallinger, Michelle Cotterchio, Graham Casey, Lonnie Zwaigenbaum, Kerstin Wittemeyer, Kirsty Wing, Simon Wallace, Herman Van Engeland, Ana Tryfon, Susanne Thomson, Latha Soorya, Bernadette Rogé, Wendy Roberts, Fritz Poustka, Susana Mouga, Nancy Minshew, L. Alison McInnes, Susan G. McGrew, Catherine Lord, Marion Leboyer, Ann S. Le Couteur, Alexander Kolevzon, Patricia Jiménez González, Suma Jacob, Richard Holt, Stephen Guter, Jonathan Green, Andrew Green, Christopher Gillberg, Bridget A. Fernandez, Frederico Duque, Richard Delorme, Geraldine Dawson, Pauline Chaste, Cátia Café, Sean Brennan, Thomas Bourgeron, Patrick F. Bolton, Sven Bölte, Raphael Bernier, Gillian Baird, Anthony J. Bailey, Evdokia Anagnostou, Joana Almeida, Ellen M. Wijsman, Veronica J. Vieland, Astrid M. Vicente, Gerard D. Schellenberg, Margaret Pericak-Vance, Andrew D. Paterson, Jeremy R. Parr, Guiomar Oliveira, John I. Nurnberger, Anthony P. Monaco, Elena Maestrini, Sabine M. Klauck, Hakon Hakonarson, Jonathan L. Haines, Daniel H. Geschwind, Christine M. Freitag, Susan E. Folstein, Sean Ennis, Hilary Coon, Agatino Battaglia, Peter Szatmari, James S. Sutcliffe, Joachim Hallmayer, Michael Gill, Edwin H. Cook, Joseph D. Buxbaum, Bernie Devlin, Louise Gallagher, Catalina Betancur, and Stephen W. Scherer. Convergence of genes and cellular pathways dysregulated in autism spectrum disorders. *American Journal of Human Genetics*, 94(5):677–694, 2014.

[152] D. D. Pollock and W. R. Taylor. Effectiveness of correlation analysis in identifying protein residues undergoing correlated evolution. *Protein engineering*, 10(6):647–657, 1997.

[153] David D. Pollock, William R. Taylor, and Nick Goldman. Coevolving Protein Residues: Maximum Likelihood Identification and Relationship to Structure. *Journal of Molecular Biology*, 287(1):187–198, 1999.

[154] David D. Pollock, Grant Thiltgen, and Richard A. Goldstein. Amino acid coevolution induces an evolutionary Stokes shift. *Proceedings of the National Academy of Sciences of the United States of America*, 109(21):E1352–E1359, 2012.

[155] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 2nd editio edition, 1992.

[156] L Pritchard, P Bladon, J M O Mitchell, and M J Dufton. Evaluation of a novel method for the identification of coevolving protein residues. *Protein engineering*, 14(8):549–55, 2001.

[157] Michael J. Quinn. *Parallel Computing: Theory and Practice*. McGraw-Hill, New York, 2 edition, 1994.

[158] R Core Team. R: A Language and Environment for Statistical Computing. `https://www.r-project.org/`, 2016.

[159] Andrew Rambaut and Nicholas C. Grassly. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Computer applications in the biosciences*, 13(3):235–238, 1997.

[160] Vasily Ramensky, Peer Bork, and Shamil Sunyaev. Human non-synonymous SNPs: server and survey. *Nucleic acids research*, 30(17):3894–3900, 2002.

[161] Nornadiah Mohd Razali and Yap Bee Wah. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33, 2011.

[162] Rebecca J. Safran and Patrik Nosil. Speciation: The Origin of New Species. `http://www.nature.com/scitable/knowledge/library/speciation-the-origin-of-new-species-26230527`. [Online; accessed 2016-11-04].

[163] Amaia Sangrador. What are protein domains? `https://www.ebi.ac.uk/training/online/course/introduction-protein-classification-ebi/protein-classification/what-are-protein-domains`. [Online; accessed 2017-05-15].

[164] Scan. MSI GeForce GTX 660 Ti Overclocked NVIDIA Graphics Card - 2GB [accessed through the Internet Archive]. `http://web.archive.org/web/20130919080957/http://www.scan.co.uk/products/2gb-msi-gtx-660-ti-oc-28nm-pcie-30-(x16)-6008mhz-gddr5-gpu-941mhz-boost-1019mhz-cores-1344`. [Online; accessed 2013-09-19].

[165] Scott Schwartz, W. James Kent, Arian Smit, Zheng Zhang, Robert Baertsch, Ross C. Hardison, David Haussler, and Webb Miller. Human-Mouse Alignments with BLASTZ. *Genome Research*, 13(1):103–107, 2003.

[166] Scott Schwartz, Zheng Zhang, Kelly A. Frazer, Arian Smit, Cathy Riemer, John Bouck, Richard Gibbs, Ross Hardison, and Webb Miller. PipMaker–A Web Server for Aligning Two Genomic DNA Sequences. *Genome Research*, 10(4):577–586, 2000.

[167] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[168] I. N. Shindyalov, N. A. Kolchanov, and C. Sander. Can three-dimensional contacts in protein structures be predicted by analysis of correlated mutations? *Protein engineering*, 7(3):349–58, 1994.

[169] Fabian Sievers, Andreas Wilm, David Dineen, Toby J. Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Soding, Julie D. Thompson, and Desmond G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1):539–539, 2011.

[170] Tarjinder Singh, Mitja I Kurki, David Curtis, Shaun M Purcell, Lucy Crooks, Jeremy McRae, Jaana Suvisaari, Himanshu Chheda, Douglas Blackwood, Gerome Breen, Olli Pietiläinen, Sebastian S Gerety, Muhammad Ayub, Moira Blyth, Trevor Cole, David Collier, Eve L Coomber, Nick Craddock, Mark J Daly, John Danesh, Marta Di-Forti, Alison Foster, Nelson B Freimer, Daniel Geschwind, Mandy Johnstone, Shelagh Joss, Georg Kirov, Jarmo Körkkö, Outi Kuismin, Peter Holmans, Christina M Hultman, Conrad Iyegbe, Jouko Lönnqvist, Minna Männikkö, Steve A McCarroll, Peter McGuffin, Andrew M McIntosh, Andrew McQuillin, Jukka S Moilanen, Carmel Moore, Robin M Murray, Ruth Newbury-Ecob, Willem Ouwehand, Tiina Paunio, Elena Prigmore, Elliott Rees, David Roberts, Jennifer Sambrook, Pamela Sklar, David St Clair, Juha Veijola, James T R Walters, Hywel Williams, Patrick F Sullivan, Matthew E Hurles, Michael C O'Donovan, Aarno Palotie, Michael J Owen, and Jeffrey C Barrett. Rare loss-of-function variants in SETD1A are associated with schizophrenia and developmental disorders. *Nature Neuroscience*, 19(4):571–577, 2016.

[171] Tarjinder Singh, James T R Walters, Mandy Johnstone, David Curtis, Jaana Suvisaari, Minna Torniainen, Elliott Rees, Conrad Iyegbe, Douglas Blackwood, Andrew M McIntosh, Georg Kirov, Daniel Geschwind, Robin M Murray, Marta Di Forti, Elvira Bramon, Michael Gandal, Christina M Hultman, Pamela Sklar, Aarno Palotie, Patrick F Sullivan, Michael C O'Donovan, Michael J Owen, and Jeffrey C Barrett. The contribution of rare variants to risk of schizophrenia in individuals with and without intellectual disability. *Nature Genetics*, 49(8):1167–1173, 2017.

[172] Stack Exchange. random - Randomize matrix in perl, keeping row and column totals the same - Stack Overflow. `http://stackoverflow.com/questions/2133268/randomize-matrix-in-perl-keeping-row-and-column-totals-the-same`. [Online; accessed 2016-03-30].

[173] Jens Stoye, Dirk Evers, and Folker Meyer. Rose: generating sequence families. *Bioinformatics (Oxford, England)*, 14(2):157–163, 1998.

[174] Cory L. Strope, Kevin Abel, Stephen D. Scott, and Etsuko N. Moriyama. Biological Sequence Simulation for Testing Complex Evolutionary Hypotheses: indel-Seq-Gen version 2.0. *Molecular biology and evolution*, 26(11):2581–93, 2009.

[175] Gürol M. Süel, Steve W. Lockless, Mark A. Wall, and Rama Ranganathan. Evolutionarily conserved networks of residues mediate allosteric communication in proteins. *Nature structural biology*, 10(1):59–69, 2003.

[176] David Talavera, Simon C. Lovell, and Simon Whelan. Covariation Is a Poor Measure of Molecular Coevolution. *Molecular Biology and Evolution*, 32(9):2456–2468, 2015.

[177] William R. Taylor and Kerr Hatrick. Compensating changes in protein multiple sequence alignments. *Protein engineering*, 7(3):341–348, 1994.

[178] Tiberiu Teileanu, Lucy J. Colwell, and Stanislas Leibler. Protein Sectors: Statistical Coupling Analysis versus Conservation. *PLOS Computational Biology*, 11:e1004091, 2015.

[179] The Hoekstra Lab. Sexual Selection and Speciation. `http://hoekstra.oeb.harvard.edu/projects-sexual-selection`. [Online; accessed 2016-11-07].

[180] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–80, 1994.

[181] Elisabeth R. M. Tillier and Thomas W. H. Lui. Using multiple interdependency to separate functional from phylogenetic correlations in protein alignments. *Bioinformatics (Oxford, England)*, 19(6):750–755, 2003.

[182] Kentaro Tomii and Minoru Kanehisa. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *"Protein Engineering, Design and Selection"*, 9(l):27–36, 1996.

[183] P. Tufféry. CS-PSeq-Gen: Simulating the evolution of protein sequence under constraints. *Bioinformatics (Oxford, England)*, 18(7):1015–1016, 2002.

[184] Pierre Tuffery and Pierre Darlu. Exploring a phylogenetic approach for the detection of correlated substitutions in proteins. *Molecular biology and evolution*, 17(11):1753–9, 2000.

[185] UCSC. Cons 7 Verts Track Settings. `http://genome.ucsc.edu/cgi-bin/hgTrackUi?db=hg38{%}5C{&}g=cons7way`.

[186] UniProt Consortium. About UniProt. `http://www.uniprot.org/help/about`. [Online; accessed 2017-04-25].

[187] Davis University of California. Protein Structure Prediction Center. `http://predictioncenter.org/`. [Online; accessed 2017-04-13].

[188] University of Utah. Genetic Linkage. `http://learn.genetics.utah.edu/content/pigeons/geneticlinkage/`.

[189] Peter J. Waddell, Hirohisa Kishino, and Rissa Ota. Phylogenetic Methodology for Detecting Protein Interactions. *Molecular Biology and Evolution*, 24(3):650–659, 2007.

[190] Michael Waechter, Kathrin Jaeger, Daniel Thuerck, Stephanie Weissgraeber, Sven Widmer, Michael Goesele, and Kay Hamacher. Using

graphics processing units to investigate molecular coevolution. *Concurrency and Computation: Practice and Experience*, 26(6):1278–1296, 2014.

[191] Michael Waechter, Kathrin Jaeger, Stephanie Weissgraeber, Sven Widmer, Michael Goesele, and Kay Hamacher. Information-Theoretic Analysis of Molecular (Co)Evolution Using Graphics Processing Units. In *ECMLS '12 Proceedings of the 3rd international workshop on Emerging computational methods for the life sciences*, pages 49–58, 2012.

[192] Iain M. Wallace, Orla O'Sullivan, Desmond G. Higgins, and Cedric Notredame. M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Research*, 34(6):1692–1699, 2006.

[193] Zhengyuan O. Wang and David D. Pollock. Coevolutionary patterns in cytochrome c oxidase subunit I depend on structural and functional context. *Journal of molecular evolution*, 65(5):485–95, 2007.

[194] Andrew M. Waterhouse, James B. Procter, David M. A. Martin, Michèle Clamp, and Geoffrey J. Barton. Jalview Version 2–a multiple sequence alignment editor and analysis workbench. *Bioinformatics*, 25(9):1189–1191, 2009.

[195] Philipp Weil, Franziska Hoffgaard, and Kay Hamacher. Estimating sufficient statistics in co-evolutionary analysis by mutual information. *Computational biology and chemistry*, 33(6):440–4, 2009.

[196] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420, 1962.

[197] Nicholas Wilt. *The CUDA handbook: a comprehensive guide to GPU programming*. Addison-Wesley, Boston, 2013.

[198] Yuri I. Wolf, Igor B. Rogozin, and Eugene V. Koonin. Coelomata and Not Ecdysozoa: Evidence from Genome-Wide Phylogenetic Analysis. *Genome Research*, 14(1):29–36, 2004.

[199] Kurt R. Wollenberg and William R. Atchley. Separation of phylogenetic and functional associations in biological sequences by using the parametric bootstrap. *Proceedings of the National Academy of Sciences of the United States of America*, 97(7):3288–3291, 2000.

[200] 'Xenonblast'. Cartoon representation of the Beta-meander motif. `https://commons.wikimedia.org/wiki/File:Beta-`

meander1.png. Accessed on 04/11/16. This is in the Public Domain.

[201] Ziheng Yang. Among-site rate variation and its impact on phylogenetic analyses. *Trends in Ecology & Evolution*, 11(9):367–372, 1996.

[202] Chen-Hsiang Yeang and David Haussler. Detecting Coevolution in and among Protein Domains. *PLoS Computational Biology*, 3(11):2122–2134, 2007.

[203] Marketa Zvelebil and Jeremy O. Baum. *Understanding Bioinformatics*. Garland Science, New York, 2008.

Appendices

A.1 Examples of using Z-score approaches

Here we give an example of calculating a Z-score using the two Z-score approaches: the shuffle Z-score approach and the original values Z-score approach. The alignment we shall be using is shown in Figure 1.

A.1.1 Shuffle

We want to calculate the Z-score for the pair of columns 2 and 5. First we calculate the correlation value (this could be any correlation value; here we choose Mutual Information (MI)), which works out as 1.5.

Next we shuffle every column in the alignment in place, to obtain the alignment shown in Figure 2. We then calculate the Mutual Information for columns 2 and 5 again, which works out as 1.

Shuffling the alignment a final time (in this example, we only shuffle the columns twice to show how the approach works), we obtain the alignment in Figure 3. The Mutual Information for columns 2 and 5 this time is 1.5.

	0	1	2	3	4	5
0	M	G	R	V	G	Y
1	G	K	P	R	A	E
2	R	G	G	A	L	G
3	N	R	G	A	G	G

Figure 1: An example alignment.

	0	1	2	3	4	5
0	G	G	G	R	G	E
1	N	G	R	A	L	Y
2	M	K	G	A	G	G
3	R	R	P	V	A	G

Figure 2: The alignment shown in Figure 1, with each column shuffled in place.

	0	1	2	3	4	5
0	M	G	G	A	L	G
1	N	R	G	V	G	G
2	G	K	R	A	A	E
3	R	G	P	R	G	Y

Figure 3: The alignment shown in Figure 2, with each column shuffled in place.

Now that we have decided to stop shuffling and re-calculating, we calculate the mean and standard deviation of the shuffle MI values. The mean of 1 and 1.5 is 1.25, and the standard deviation is 0.25. The Z-score for the pair 2 and 5 is:

$$Z_{2,5} = \frac{1.5 - 1.25}{0.25} = 1 \tag{1}$$

The 1.5 in the calculation is the original value of the MI for positions 2 and 5 before shuffling, the 1.25 is the mean of the MI values after shuffling, and the 0.25 is the standard deviation of the MI values after shuffling.

A.1.2 Original values

We want to calculate the Z-score for the pair of columns 2 and 5. To do so, we actually need to calculate the correlation values (Mutual Information in this case) for all pairs of positions. These are:

- $MI_{0,1}$ = 1.5
- $MI_{0,2}$ = 1.5
- $MI_{0,3}$ = 1.5
- $MI_{0,4}$ = 1.5
- $MI_{0,5}$ = 1.5
- $MI_{1,2}$ = 1
- $MI_{1,3}$ = 1
- $MI_{1,4}$ = 1
- $MI_{1,5}$ = 1
- $MI_{2,3}$ = 1.5
- $MI_{2,4}$ = 1
- $MI_{2,5}$ = 1.5
- $MI_{3,4}$ = 1.5
- $MI_{3,5}$ = 1
- $MI_{4,5}$ = 1.5

We calculate the mean and the standard deviation of all the correlation values, which works out as 1.3 and 0.2449 respectively. The Z-score for positions 2 and 5 can now be calculated as:

$$Z_{2,5} = \frac{1.5 - 1.3}{0.2449} = 0.8167 \tag{2}$$

The 1.5 in the equation is the correlation calculated for the pair 2 and 5, the 1.3 is the mean of all of the correlations, and the 0.2449 is the standard deviation of all of the correlations.

A.2 Co-evolving cluster choices

To decide the group sizes and number of groups of co-evolving sites to use in our simulations, we considered the co-evolving groups detected in real data. A summary of all papers considered (in note form) is below:

- [16]: three proteins (or families) studied:
  - PDZ: three clusters of size 2, three clusters of size 3, one cluster of size 4, one cluster of size 5
  - CS: 26 clusters of size 2, 11 clusters of size 3, five clusters of size 4, one cluster of size 5, one cluster of size 9
  - GPCR: 16 clusters of size 2, three clusters of size 3, two clusters of size 4, two clusters of size 5, one cluster of size 10
- [24]: pairs technically, but they were interested in the number of positions each position was "connected" to, and categorised the number of connections into two bins, $<= 10$ and > 10; Figure 6 shows the connections for three protein families:
  - Maf_Ham1: 87 connections between 15 sites (number of connections taken from text)
  - Pterin binding enzymes: 15 sites
  - LacI/PurR: 25 sites
- [25]: pairs that can make networks. Figures 3 and 6 show examples for seven protein families:
  - GlnRS catalytic domain: one cluster of size 10
  - Ferritin-like diiron-carboxylate protein domain: one cluster of size 7
  - YjgF_YER057c_UK114_family: two clusters of size 2, one cluster of size 3, one cluster of size 4
  - CAP family of transcription factors: four clusters of size 2, one cluster of size 5
  - Hedgehog/intein domains: three clusters of size 2, one cluster of size 8
  - Phosphoglycerate kinase: 10 clusters of size 2, four clusters of size 3
  - PAL and HAL domain: three clusters of size 2, one cluster of size 7, one cluster of size 27

- [43]: pairs and groups of up to 10; for three data sets:
  - Myoglobin data set: six clusters of size 2, three clusters of size 3, four clusters of size 4, two clusters of size 5, two clusters of size 7
  - SRK data set: seven clusters of size 2, two clusters of size 3, one cluster of size 4, one cluster of size 5, one cluster of size 6, one cluster of size 7, one cluster of size 9
  - MAP data set: 30 clusters of size 2, seven clusters of size 3, two clusters of size 4, one cluster of size 5, three clusters of size 9
- [46]: mostly pairs; one group of three (one site has two predicted compensatory sites)
- [73]: mention of residues with Z-scores over a threshold of 4 with three or more other residues (but no more detail than that). Network of 16 connected together in Figure 4; cannot obtain file with grouped residues (not available via Wayback Machine)
- [74]: pairs but groups also. Groups are five segments: 27-29 (3 residues), 70-72 (3 residues), 80-82 (3 residues), 122-124 (3 residues), and 144-149 (6 residues); these are contiguous sections (e.g. 3 residues in a line) due to how they looked at the pairwise values using a sliding window
- [75]: one cluster of 7 sites given as an example
- [84]: sectors across five protein families:
  - S1A: two clusters of size 20, one cluster of size 25
  - PDZ: one cluster of size 6, one cluster of size 10
  - PAS: one cluster of size 10, one cluster of size 17
  - SH2: one cluster of size 12, one cluster of size 13, one cluster of size 17
  - SH3: one cluster of size 5, one cluster of size 6
- [104]: there are groups; degree of individual sites can go up to about 100 (but most do not get this far) – smaller degrees tend to have higher frequencies (Figure 7)
- [110]: sectors across four protein families:
  - S1A: one cluster of size 22, one cluster of size 23, one cluster of size 25
  - PDZ: one cluster of size 6, one cluster of size 12

  - Hsp70/110: one cluster of size 11, one cluster of size 14
  - G protein: one cluster of size 26, one cluster of size 67

- [116]: pairs only, but also positions that had high Z-scores with more than one other position (no further detail than that)

- [121]: each position k is given a score, conn(k), which tells us how many residues k is in a high-scoring pair with. For example, for the protein with PFAM entry PF01053, conn(388) = 10. The 10 positions that contribute to this conn value are:

  - conn(267) = 1 or 2
  - conn(299) = 1 or 2
  - conn(86) = 3 or 4
  - conn(308) = 3 or 4
  - conn(402) = 3 or 4
  - conn(305) = 5 or 6
  - conn(386) = 5 or 6
  - conn(391) = 5 or 6
  - conn(393) = 5 or 6
  - conn(268) = 7 or 8

  (Not sure whether this was just done as an experiment.)

  - Filtering PF00018 (which contains 3506 sequences) using default filter values (to give us 471 sequences), gives us the following conn(k) values $>= 2$: nine of conn(k) = 2, two of conn(k) = 3, two of conn(k) = 4, two of conn(k) = 5, one of conn(k) = 6, one of conn(k) = 7, and one of conn(k) = 13
  - For the ATP synthase unit of E. coli we get conn(k) up to 5, 6, 7, and 13
  - For an MSA of globin sequences compiled previously, 32 positions are predicted to be in a network; in this study, nine positions had conn(k) $>= 4$; only three predictions were in agreement with the previous study
  - For TrpA largest value was 8, same for TrpB. For TrpB, five positions had conn(k) $>= 4$. TrpD had a conn(k) = 11, and five positions had conn(k) $>= 4$

  - For 1QGN, we had some conn(k) values 4, 5 (four of these), 8, and 10
  - (The following is from the supplementary information)
  - PF000026: two of conn(k) = 5, one of conn(k) = 6
  - PF000067: two of conn(k) = 5, one of conn(k) = 8, two of conn(k) = 9
  - PF00068: four of conn(k) = 5, two of conn(k) = 6, one of conn(k) = 9
  - PF000089: two of conn(k) = 5, one of conn(k) = 7, one of conn(k) = 13
  - PF00141: three of conn(k) = 5, two of conn(k) = 6, one of conn(k) = 8, three of conn(k) = 9, one of conn(k) = 10
  - PF00248: four of conn(k) = 5, one of conn(k) = 6
  - PF00331: two of conn(k) = 5, one of conn(k) = 7
  - PF00550: one of conn(k) = 5, one of conn(k) = 6, one of conn(k) = 11, one of conn(k) = 14
  - PF00557: four of conn(k) = 4, one of conn(k) = 6, one of conn(k) = 7, one of conn(k) = 9
  - PF05199: one of conn(k) = 9, one of conn(k) = 13

- [141]: mentions "contact networks", but does not seem to give any examples of these
- [178] two protein (families?):
  - PDZ: one cluster of 21 positions
  - Dihydrofolate reductase: one cluster of 14 positions (I think?)
- [191]: 2-point MI (pairs) and 3-point MI (groups of three) are investigated; they do not go into detail as to how many of these they found though
- [190]: like the above, does not go into detail

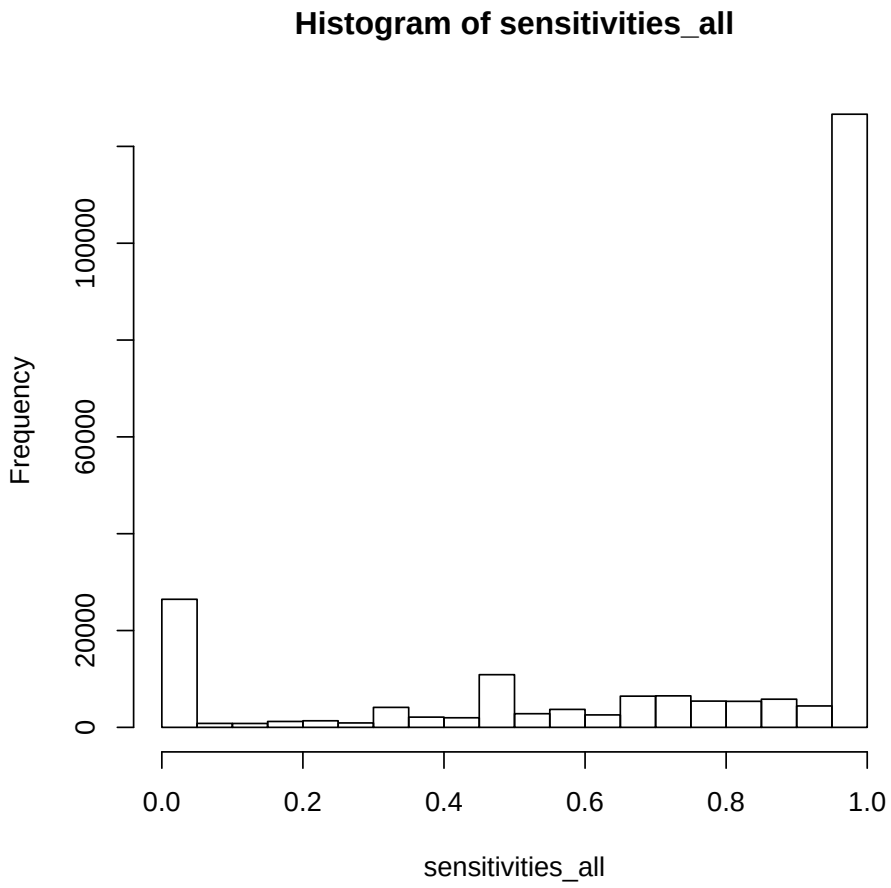

Figure 4: Histogram of all sensitivities results. This distribution is clearly not Normal.

A.3 Statistical test choice for comparison of Z-score types

To work out the type of statistical test we can use to compare the different groups of data (original values Z-score type sensitivities with shuffle Z-score type sensitivities, etc.), we must decide if our data is Normal or non-Normal. Visualising our data as a series of histograms (see Figure 4, Figure 5, and Figure 6 respectively), we see that all three data sets (sensitivities, specificities, and precisions) are not Normal.

It is good practice to not simply rely on such graphical representations of the distributions, but to also use normality tests to "prove" that the data is Normal or not Normal. Ghasemi and Zahediasl [71] recommended the Shapiro-Wilk test, and this test was also proved to have the highest power overall by Razali and Wah [161] (when comparing Shapiro-Wilk to Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling). However, we cannot use the Shapiro-Wilk test in R when the sample size is greater than 5000, which is

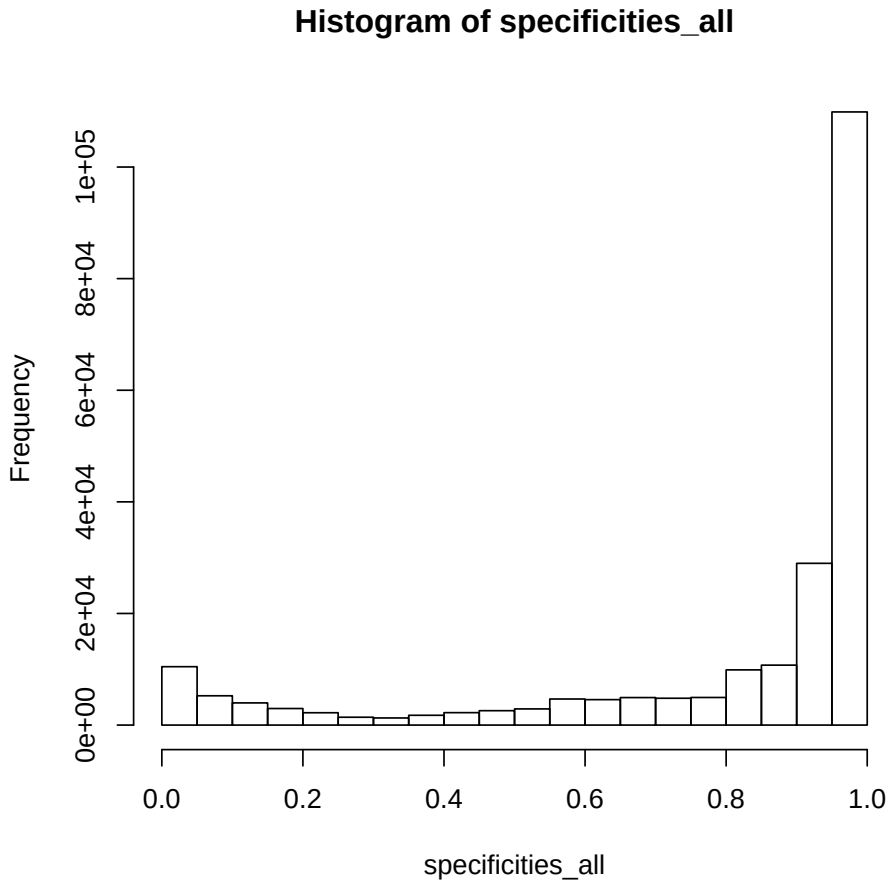

Figure 5: Histogram of all specificities results. This distribution is clearly not Normal.

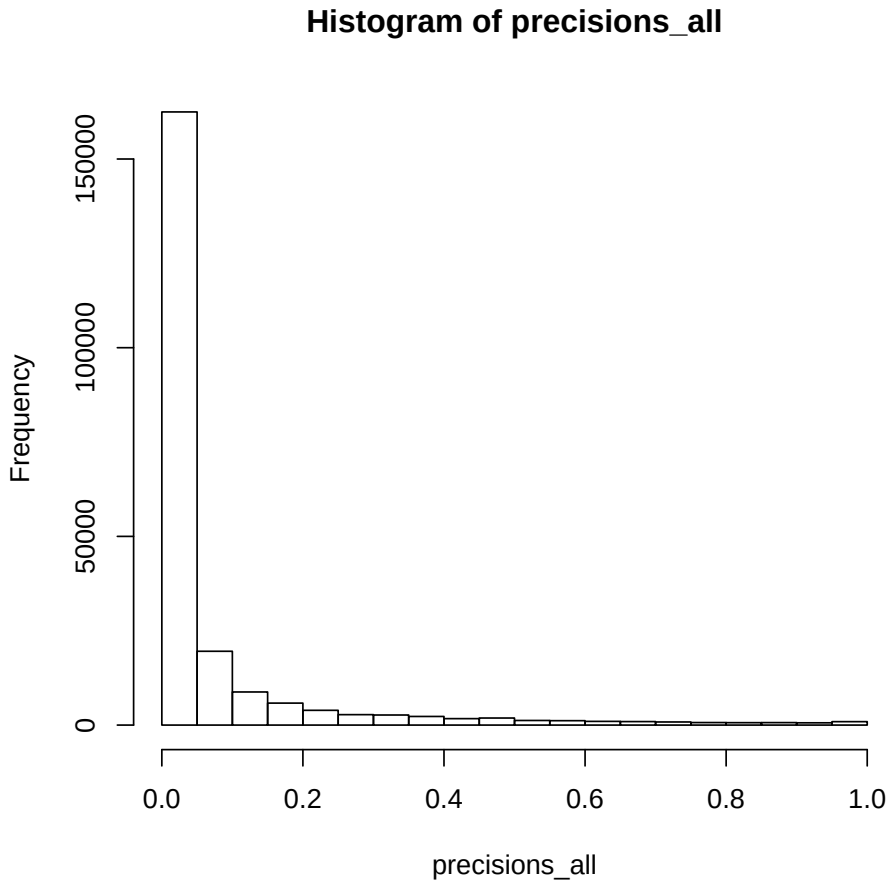

Figure 6: Histogram of all precisions results. This distribution is clearly not Normal.

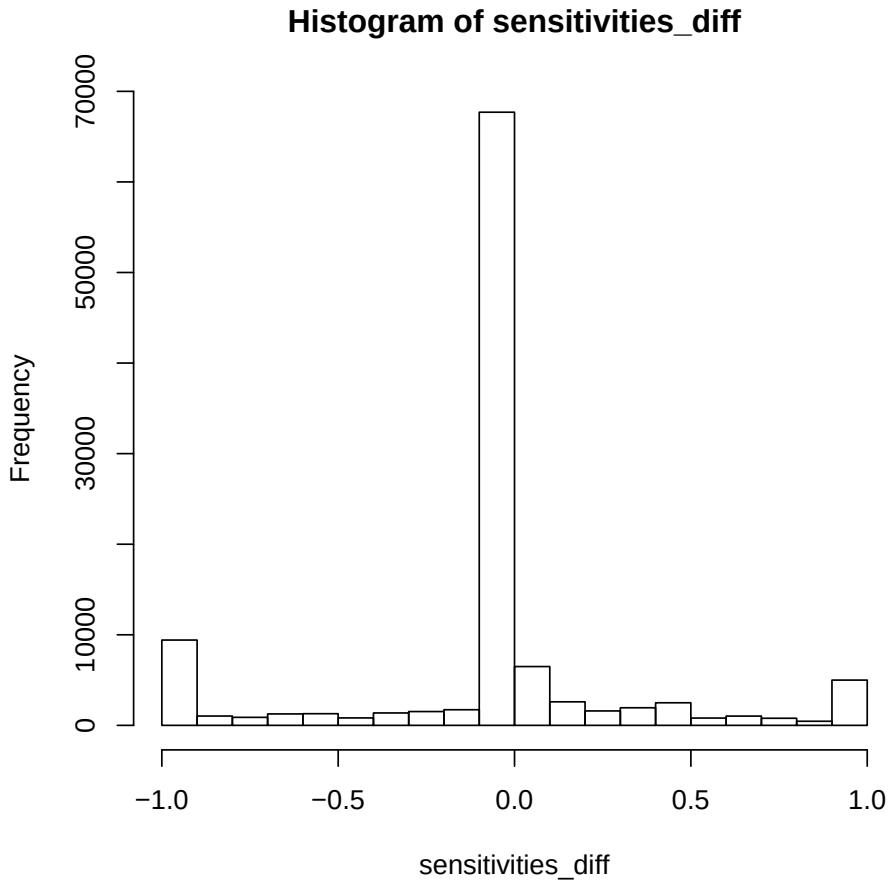

Figure 7: Histogram of (original values Z-score sensitivities) - (shuffle Z-score sensitivities) for all results.

the case for these data sets. Therefore we decided to use Anderson-Darling, as it consistently came second in the ranking of the four normality tests being compared by Razali and Wah [161]. This test can be found in the `nortest` package in R with the function `ad.test()`. Running this test for each of the data sets of all sensitivities, all specificities, and all precisions, we obtain the p-value $<$ 2.2 e-16 in all cases, which means that as we suspected, none of these distributions are Normal.

Now that we have shown that the datasets are all non-Normal, we know that we need to use a non-parametric test to compare the groups. To help us decide which statistic to use, we refer to a table compiled by UCLA designed for such a purpose [94]. Let us start with comparing the sensitivities for the original values Z-score type against the sensitivities for the shuffle Z-score type (we would use the same test for the specificities and precisions). There is one dependent variable (sensitivity), and one independent variable (Z-score type). The Z-score type can be split into two categories (called 'levels' in UCLA's table). These can be thought of as matched groups, as although the

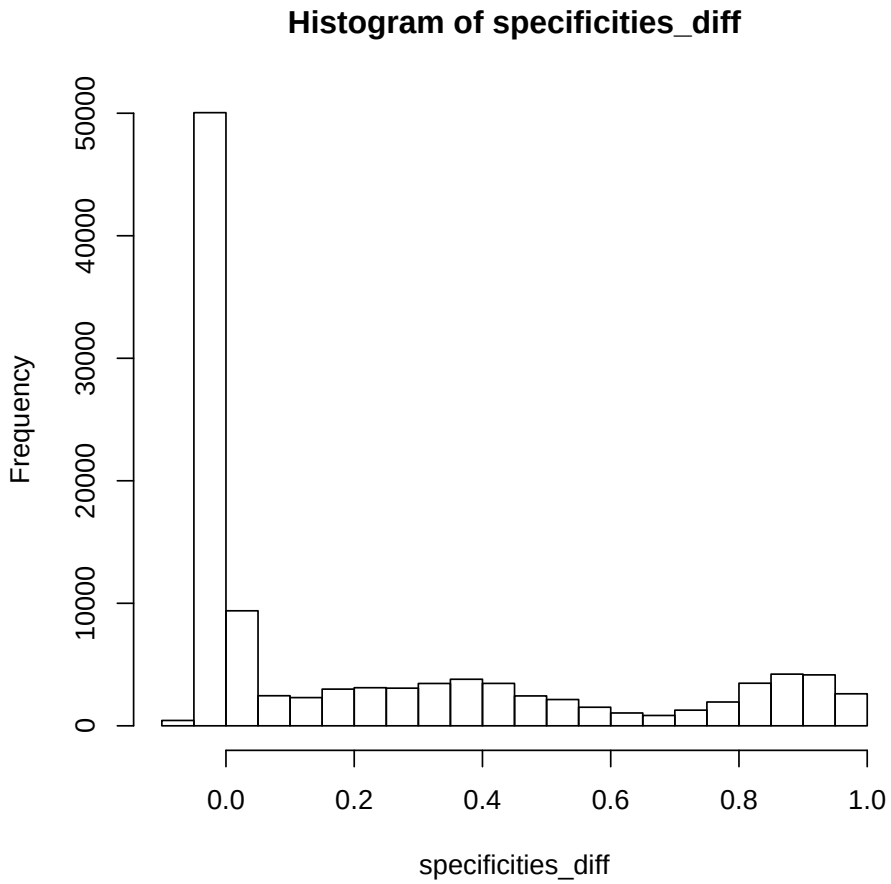

Figure 8: Histogram of (original values Z-score specificities) - (shuffle Z-score specificities) for all results.

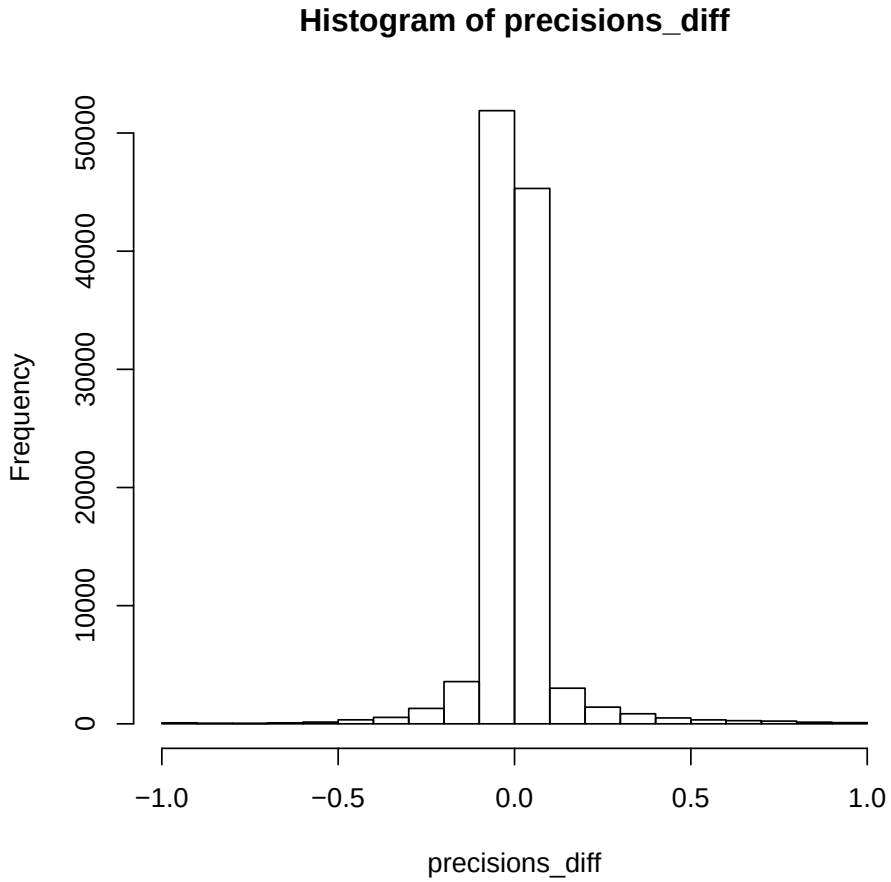

Figure 9: Histogram of (original values Z-score precisions) - (shuffle Z-score precisions) for all results.

Dataset	Mean	Median	Mode	Symmetrical?
Sensitivities	-0.0365	0	0	Could be
Specificities	0.2343	0.0082	0	Probably not
Precisions	0.0059	0	0	Probably

Table 1: The mean, median, and mode for each of the three groups of differences (between original values Z-score results and shuffle Z-score results).

two types of Z-score method are separate, the two groups we are comparing are the results of using two different Z-score methods on the same data set. The dependent variable is a continuous value (sensitivity), which fits into the 'ordinal or interval' category, giving us the Wilcoxon signed-ranks test. But, if we refer to [106], there are three assumptions we must satisfy in order to use this test. The first two relate to the type of data for the dependent and independent variables, and have already been established. The third assumption means that the differences between the two groups of points, when plotted, should look symmetrical. To establish whether this was true or not, we plotted the differences for each of the three groups (see Figure 7, Figure 8, and Figure 9).

We also calculated the mean, median, and mode for each of the three groups of differences (in a symmetrical distribution, we would expect these values to be equal). We obtained the results shown in Table 1. For the symmetrical distributions, we can use the Wilcoxon signed-rank test (with alternative="greater", meaning we expect the original values Z-score to do better than the shuffle Z-score), but for the non-symmetrical distributions, we must instead use the sign test [106]. Because the sensitivities differences, for example, are inconclusive as to whether they are symmetrical or not, we decided to go ahead and calculate the result of both tests for all groups.

A.4 Which UAA threshold does each co-evolution detection method perform best for?

Here we provide the p-values for the (one-sided) Wilcoxon rank-sum test calculations between each UAA threshold, for each co-evolution detection method. The inequality signs are used to show which UAA threshold performed better than the other; for example, for CAPS precisions, UAA threshold $2 > 3$, which means that for CAPS, we achieved higher precisions overall for UAA threshold 2 than for threshold 3. If the p-value is listed as "no difference", then this means that the p-value for the two-sided test was not significant.

A.4.1 CAPS

Precisions

UAA thresholds	p-value
$2 > 3$	0.00
$2 > 4$	0.00
$2 > 5$	$1.80 \cdot 10^{-156}$
$3 > 4$	$2.39 \cdot 10^{-38}$
$3 > 5$	$2.65 \cdot 10^{-32}$
$4 > 5$	0.00

Sensitivities

UAA thresholds	p-value
$2 > 3$	$4.56 \cdot 10^{-161}$
$2 > 4$	$2.76 \cdot 10^{-129}$
$2 > 5$	$4.18 \cdot 10^{-56}$
$3 > 4$	$3.82 \cdot 10^{-10}$
$3 > 5$	$3.66 \cdot 10^{-07}$
4 and 5	no difference

Specificities

UAA thresholds	p-value
$2 < 3$	$1.66 \cdot 10^{-08}$
$2 < 4$	$2.99 \cdot 10^{-16}$
$2 < 5$	$1.99 \cdot 10^{-09}$
$3 < 4$	0.00
$3 < 5$	0.00
4 and 5	no difference

A.4.2 PlotCorr

Precisions

UAA thresholds	p-value
$2 > 3$	$5.85 \cdot 10^{-182}$
$2 > 4$	$4.51 \cdot 10^{-265}$
$2 > 5$	$1.62 \cdot 10^{-162}$
$3 > 4$	$7.90 \cdot 10^{-56}$
$3 > 5$	$9.66 \cdot 10^{-63}$
$4 > 5$	$1.07 \cdot 10^{-09}$

Sensitivities

UAA thresholds	p-value
$2 > 3$	$1.40 \cdot 10^{-38}$
$2 > 4$	$1.18 \cdot 10^{-08}$
2 and 5	no difference
$3 < 4$	0.00
$3 < 5$	$1.10 \cdot 10^{-07}$
$4 < 5$	0.00

Specificities

UAA thresholds	p-value
$2 > 3$	$5.46 \cdot 10^{-94}$
$2 > 4$	$6.18 \cdot 10^{-94}$
$2 > 5$	$3.05 \cdot 10^{-44}$
$3 > 4$	$2.00 \cdot 10^{-10}$
$3 > 5$	$1.08 \cdot 10^{-07}$
4 and 5	no difference

A.4.3 MI

Precisions

UAA thresholds	p-value
$2 > 3$	$4.96 \cdot 10^{-55}$
$2 > 4$	$1.76 \cdot 10^{-25}$
$2 > 5$	$5.21 \cdot 10^{-11}$
3 and 4	no difference
3 and 5	no difference
4 and 5	no difference

Sensitivities

UAA thresholds	p-value
2 and 3	no difference
$2 < 4$	$1.26 \cdot 10^{-62}$
$2 < 5$	$3.13 \cdot 10^{-94}$
$3 < 4$	$2.25 \cdot 10^{-53}$
$3 < 5$	$8.05 \cdot 10^{-91}$
$4 < 5$	$4.09 \cdot 10^{-29}$

Specificities

UAA thresholds	p-value
$2 > 3$	$1.04 \cdot 10^{-47}$
$2 > 4$	$2.00 \cdot 10^{-44}$
$2 > 5$	$1.67 \cdot 10^{-21}$
$3 > 4$	$3.11 \cdot 10^{-05}$
$3 > 5$	0.00
4 and 5	no difference

A.4.4 Waddell – Kappa

Precisions

UAA thresholds	p-value
$2 > 3$	$8.87 \cdot 10^{-107}$
$2 > 4$	$5.76 \cdot 10^{-163}$
$2 > 5$	$2.38 \cdot 10^{-114}$
$3 > 4$	$5.01 \cdot 10^{-42}$
$3 > 5$	$3.54 \cdot 10^{-58}$
$4 > 5$	$1.25 \cdot 10^{-12}$

Sensitivities

UAA thresholds	p-value
$2 < 3$	0.00
$2 < 4$	$2.98 \cdot 10^{-283}$
$2 < 5$	$5.69 \cdot 10^{-137}$
$3 < 4$	$1.77 \cdot 10^{-32}$
$3 < 5$	$1.16 \cdot 10^{-31}$
$4 < 5$	$3.65 \cdot 10^{-06}$

Specificities

UAA thresholds	p-value
$2 > 3$	$3.05 \cdot 10^{-228}$
$2 > 4$	$4.49 \cdot 10^{-234}$
$2 > 5$	$2.39 \cdot 10^{-111}$
$3 > 4$	$2.79 \cdot 10^{-24}$
$3 > 5$	$1.25 \cdot 10^{-17}$
4 and 5	no difference

A.4.5 Waddell – MI

Precisions

UAA thresholds	p-value
$2 > 3$	$2.98 \cdot 10^{-100}$
$2 > 4$	$2.89 \cdot 10^{-128}$
$2 > 5$	$2.37 \cdot 10^{-96}$
$3 > 4$	$1.12 \cdot 10^{-21}$
$3 > 5$	$2.78 \cdot 10^{-38}$
$4 > 5$	$3.08 \cdot 10^{-11}$

Sensitivities

UAA thresholds	p-value
$2 < 3$	$9.48 \cdot 10^{-98}$
$2 < 4$	$1.68 \cdot 10^{-227}$
$2 < 5$	$1.24 \cdot 10^{-122}$
$3 < 4$	$2.77 \cdot 10^{-67}$
$3 < 5$	$2.99 \cdot 10^{-55}$
$4 < 5$	$2.01 \cdot 10^{-08}$

Specificities

UAA thresholds	p-value
$2 > 3$	$1.58 \cdot 10^{-169}$
$2 > 4$	$8.21 \cdot 10^{-171}$
$2 > 5$	$2.12 \cdot 10^{-79}$
$3 > 4$	$1.50 \cdot 10^{-16}$
$3 > 5$	$4.62 \cdot 10^{-12}$
4 and 5	no difference

	Sensitivity	Specificity	Precision
Z = 0			
CAPS	0.9378	0.6979	0.0329
PlotCorr	0.9375	0.8071	0.0540
Z = 1			
CAPS	0.7169	0.8878	0.0392
PlotCorr	0.9150	0.9036	0.0741
Z = 2			
CAPS	0.5168	0.9679	0.0379
PlotCorr	0.8748	0.9609	0.1157
Z = 3			
CAPS	0.4385	0.9824	0.0297
PlotCorr	0.7513	0.9839	0.1351

Table 2: The mean sensitivity, specificity, and precision values for CAPS and PlotCorr, for unique amino acid threshold 2.

	Sensitivity	Specificity	Precision
Z = 0			
CAPS	0.9582	0.5486	0.0105
PlotCorr	0.9807	0.7086	0.0168
Z = 1			
CAPS	0.5357	0.8662	0.0115
PlotCorr	0.9272	0.8646	0.0322
Z = 2			
CAPS	0.2386	0.9815	0.0052
PlotCorr	0.8337	0.9540	0.0687
Z = 3			
CAPS	0.1779	0.9916	0.0026
PlotCorr	0.5592	0.9865	0.0641

Table 3: The mean sensitivity, specificity, and precision values for CAPS and PlotCorr, for unique amino acid threshold 3.

A.5 Choosing a "lower" UAA threshold

There are a large number of co-evolving blocks for both unique amino acid thresholds 2 and 3, so to choose which to use for our "lower" UAA (Unique Amino Acid) threshold, we shall compare the performance of CAPS and PlotCorr at each (these two methods perform the best at lower thresholds).

We group the sensitivities, specificities, and precisions for the CAPS and PlotCorr methods for the original values Z-score and Z-score thresholds 0, 1, 2, and 3, separately for unique amino acid thresholds 2 and 3 (so we have two sets of sensitivities, two of specificities, and two for precisions). The test we shall use is the Wilcoxon rank sum test ("Wilcoxon-Mann-Whitney"); this is the same as the Wilcoxon signed-rank test but for use with unpaired data.

From looking at the means in Table 2 and Table 3, it appears that the results for unique amino acid 2 threshold are better than those for the threshold 3 (again, we are more interested in the performance in terms of specificity and precision values). Using the Wilcoxon rank sum test with alternative="greater", the p-values obtained are all significant (p-value $< 2.2e-16$ for sensitivity and precision, p-value = $1.466e{-}14$ for specificity). This means that the results for unique amino acid threshold 2 are indeed better than for unique amino acid threshold 3.

A.6 Results by number of sequences p-values

Here we present the p-values of the one-sided Wilcoxon signed-rank and Sign tests for the results when examining the number of sequences in an alignment. If "no difference" is present in one column, then that means that no difference was found between the groups for that test. If "no difference" is present in both columns, then the initial one-sided test was not able to establish a difference between the two groups being compared. The values in the tables are set to show 2 decimal places, so '0.00' is likely to be a number not small enough to be displayed as $\mathrm{X} \cdot 10^{-\mathrm{Y}}$.

A.6.1 Comparing co-evolution detection methods

In this section, we compare the performance of pairs of methods for each number of sequences, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

6 sequences

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.62 \cdot 10^{-200}$	0.00
MI > CAPS	$1.05 \cdot 10^{-142}$	$1.31 \cdot 10^{-222}$
Waddell – Kappa > CAPS	$8.40 \cdot 10^{-152}$	$2.15 \cdot 10^{-189}$
CAPS > Waddell – MI	$4.94 \cdot 10^{-52}$	$1.11 \cdot 10^{-16}$
PlotCorr > MI	$3.14 \cdot 10^{-199}$	$2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$5.05 \cdot 10^{-48}$	$1.57 \cdot 10^{-69}$
PlotCorr > Waddell – MI	$1.85 \cdot 10^{-198}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$1.34 \cdot 10^{-133}$	$1.57 \cdot 10^{-171}$
MI > Waddell – MI	$8.19 \cdot 10^{-171}$	$2.22 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$3.66 \cdot 10^{-157}$	$2.22 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
CAPS and PlotCorr	no difference	no difference
MI > CAPS	0.02	$5.58 \cdot 10^{-06}$
CAPS > Waddell – Kappa	$6.66 \cdot 10^{-111}$	$-2.22 \cdot 10^{-16}$
CAPS and Waddell – MI	no difference	no difference
MI > PlotCorr	$1.80 \cdot 10^{-13}$	$8.47 \cdot 10^{-22}$
PlotCorr > Waddell – Kappa	$1.11 \cdot 10^{-109}$	$-2.22 \cdot 10^{-16}$
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$2.46 \cdot 10^{-113}$	$-2.22 \cdot 10^{-16}$
MI > Waddell – MI	$1.01 \cdot 10^{-09}$	$5.73 \cdot 10^{-06}$
Waddell – MI > Waddell – Kappa	$3.13 \cdot 10^{-103}$	$1.84 \cdot 10^{-186}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$7.82 \cdot 10^{-205}$	0.00
MI > CAPS	$2.07 \cdot 10^{-183}$	$2.14 \cdot 10^{-228}$
Waddell – Kappa > CAPS	$3.97 \cdot 10^{-205}$	0.00
CAPS > Waddell – MI	$9.06 \cdot 10^{-53}$	$1.11 \cdot 10^{-16}$
PlotCorr > MI	$8.00 \cdot 10^{-204}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$9.02 \cdot 10^{-202}$	0.00
PlotCorr > Waddell – MI	$4.47 \cdot 10^{-200}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$3.97 \cdot 10^{-205}$	0.00
MI > Waddell – MI	$1.96 \cdot 10^{-149}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$2.60 \cdot 10^{-204}$	$1.11 \cdot 10^{-16}$

10 sequences

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$5.72 \cdot 10^{-228}$	0.00
MI > CAPS	$3.09 \cdot 10^{-54}$	$3.22 \cdot 10^{-99}$
Waddell – Kappa > CAPS	$1.12 \cdot 10^{-209}$	$1.72 \cdot 10^{-312}$
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$2.25 \cdot 10^{-239}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$6.76 \cdot 10^{-104}$	$2.01 \cdot 10^{-173}$
PlotCorr > Waddell – MI	$1.25 \cdot 10^{-200}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$1.52 \cdot 10^{-203}$	$1.07 \cdot 10^{-299}$
MI > Waddell – MI	$2.09 \cdot 10^{-75}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$8.15 \cdot 10^{-225}$	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	no difference	0.00
MI > CAPS	$4.61 \cdot 10^{-10}$	$3.36 \cdot 10^{-19}$
CAPS > Waddell – Kappa	$6.77 \cdot 10^{-113}$	0.00
CAPS > Waddell – MI	$1.54 \cdot 10^{-46}$	0.00
MI > PlotCorr	$7.64 \cdot 10^{-26}$	$2.24 \cdot 10^{-44}$
PlotCorr > Waddell – Kappa	$3.13 \cdot 10^{-113}$	$-2.22 \cdot 10^{-16}$
PlotCorr > Waddell – MI	$9.30 \cdot 10^{-40}$	0.00
MI > Waddell – Kappa	$1.66 \cdot 10^{-121}$	$-2.22 \cdot 10^{-16}$
MI > Waddell – MI	$1.82 \cdot 10^{-57}$	0.00
Waddell – MI > Waddell – Kappa	$1.65 \cdot 10^{-65}$	$3.17 \cdot 10^{-117}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.46 \cdot 10^{-246}$	0.00
MI > CAPS	$2.14 \cdot 10^{-136}$	$8.88 \cdot 10^{-96}$
Waddell – Kappa > CAPS	$4.54 \cdot 10^{-250}$	0.00
Waddell – MI > CAPS	$5.22 \cdot 10^{-30}$	$7.43 \cdot 10^{-05}$
PlotCorr > MI	$2.64 \cdot 10^{-249}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$2.95 \cdot 10^{-247}$	0.00
PlotCorr > Waddell – MI	$8.60 \cdot 10^{-216}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$4.04 \cdot 10^{-250}$	0.00
MI > Waddell – MI	$8.21 \cdot 10^{-19}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$1.81 \cdot 10^{-249}$	$1.11 \cdot 10^{-16}$

22 sequences

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	0.00	0.00
MI > CAPS	$3.39 \cdot 10^{-28}$	$8.71 \cdot 10^{-46}$
Waddell – Kappa > CAPS	0.00	0.00
Waddell – MI > CAPS	$4.19 \cdot 10^{-204}$	0.00
PlotCorr > MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	0.00	0.00
PlotCorr > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	0.00	0.00
Waddell – MI > MI	$7.52 \cdot 10^{-90}$	$4.93 \cdot 10^{-107}$
Waddell – Kappa > Waddell – MI	0.00	$2.22 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.09 \cdot 10^{-218}$	$4.94 \cdot 10^{-198}$
CAPS > MI	0.00	$8.01 \cdot 10^{-05}$
Waddell – Kappa > CAPS	$6.34 \cdot 10^{-113}$	$9.23 \cdot 10^{-56}$
Waddell – MI > CAPS	$4.03 \cdot 10^{-210}$	$8.79 \cdot 10^{-170}$
PlotCorr > MI	$6.36 \cdot 10^{-262}$	$1.11 \cdot 10^{-16}$
PlotCorr > Waddell – Kappa	$6.53 \cdot 10^{-66}$	$1.11 \cdot 10^{-16}$
Waddell – MI > PlotCorr	$1.86 \cdot 10^{-13}$	$1.29 \cdot 10^{-21}$
Waddell – Kappa > MI	$6.67 \cdot 10^{-156}$	$1.22 \cdot 10^{-154}$
Waddell – MI > MI	$3.10 \cdot 10^{-281}$	0.00
Waddell – MI > Waddell – Kappa	$1.01 \cdot 10^{-164}$	$2.99 \cdot 10^{-300}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.71 \cdot 10^{-197}$	$1.20 \cdot 10^{-41}$
MI > CAPS	no difference	$6.56 \cdot 10^{-13}$
Waddell – Kappa > CAPS	$8.24 \cdot 10^{-222}$	$1.80 \cdot 10^{-48}$
CAPS > Waddell – MI	$6.31 \cdot 10^{-57}$	0.01
PlotCorr > MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	0.00	0.00
PlotCorr > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	0.00	0.00
MI > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$

"Higher" UAA threshold

6 sequences

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	0.00	0.00
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	0.00	0.00
Waddell – MI > CAPS	0.00	0.00
PlotCorr > MI	0.00	0.00
Waddell – Kappa > PlotCorr	0.00	0.00
Waddell – MI > PlotCorr	0.00	0.00
Waddell – Kappa > MI	0.00	0.00
Waddell – MI > MI	0.00	0.00
Waddell – Kappa and Waddell – MI	no difference	no difference

Sensitivities – no differences

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	0.01	0.00
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	0.00	0.00
Waddell – MI > CAPS	0.00	0.00
PlotCorr > MI	0.01	0.00
Waddell – Kappa > PlotCorr	0.00	0.00
Waddell – MI > PlotCorr	0.00	0.00
Waddell – Kappa > MI	0.00	0.00
Waddell – MI > MI	0.00	0.00
Waddell – Kappa and Waddell – MI	no difference	no difference

10 sequences

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
MI > CAPS	0.00	0.00
Waddell – Kappa > CAPS	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > CAPS	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
PlotCorr > MI	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – Kappa > PlotCorr	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > PlotCorr	0.00	0.00
Waddell – Kappa > MI	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > MI	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > Waddell – Kappa	0.03	0.01

Sensitivities – no differences

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
MI > CAPS	$3.81 \cdot 10^{-05}$	0.00
Waddell – Kappa > CAPS	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > CAPS	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
PlotCorr > MI	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – Kappa > PlotCorr	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > PlotCorr	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – Kappa > MI	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > MI	$7.63 \cdot 10^{-06}$	$7.63 \cdot 10^{-06}$
Waddell – MI > Waddell – Kappa	0.01	0.00

22 sequences

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.30 \cdot 10^{-102}$	$5.11 \cdot 10^{-162}$
MI > CAPS	$6.72 \cdot 10^{-104}$	$1.70 \cdot 10^{-175}$
Waddell – Kappa > CAPS	$8.02 \cdot 10^{-105}$	$7.89 \cdot 10^{-167}$
Waddell – MI > CAPS	$2.65 \cdot 10^{-110}$	$8.90 \cdot 10^{-180}$
PlotCorr > MI	$4.74 \cdot 10^{-92}$	0.00
Waddell – Kappa > PlotCorr	$1.55 \cdot 10^{-89}$	$6.78 \cdot 10^{-143}$
Waddell – MI > PlotCorr	$2.00 \cdot 10^{-87}$	$6.58 \cdot 10^{-144}$
Waddell – Kappa > MI	$4.55 \cdot 10^{-99}$	$1.59 \cdot 10^{-137}$
Waddell – MI > MI	$4.68 \cdot 10^{-107}$	$2.51 \cdot 10^{-165}$
Waddell – Kappa > Waddell – MI	$6.76 \cdot 10^{-29}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$9.40 \cdot 10^{-49}$	$2.46 \cdot 10^{-41}$
MI > CAPS	$2.83 \cdot 10^{-79}$	$4.25 \cdot 10^{-118}$
Waddell – Kappa > CAPS	$6.24 \cdot 10^{-47}$	$2.55 \cdot 10^{-38}$
Waddell – MI > CAPS	$5.83 \cdot 10^{-56}$	$2.93 \cdot 10^{-46}$
MI > PlotCorr	$4.23 \cdot 10^{-41}$	$1.01 \cdot 10^{-68}$
PlotCorr and Waddell – Kappa	no difference	no difference
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$1.99 \cdot 10^{-35}$	0.00
MI > Waddell – MI	$3.47 \cdot 10^{-26}$	0.00
Waddell – MI > Waddell – Kappa	$2.40 \cdot 10^{-15}$	$5.56 \cdot 10^{-21}$

Specificities

Methods	Wilcoxon p-value	Sign p-value
PlotCorr > CAPS	$2.63 \cdot 10^{-41}$	0.00
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	$1.47 \cdot 10^{-45}$	0.00
Waddell – MI > CAPS	$4.85 \cdot 10^{-45}$	0.00
PlotCorr > MI	$4.20 \cdot 10^{-122}$	0.00
Waddell – Kappa > PlotCorr	$4.07 \cdot 10^{-122}$	$1.11 \cdot 10^{-221}$
Waddell – MI > PlotCorr	$1.02 \cdot 10^{-121}$	$4.45 \cdot 10^{-203}$
Waddell – Kappa > MI	$4.07 \cdot 10^{-122}$	$1.11 \cdot 10^{-221}$
Waddell – MI > MI	$4.07 \cdot 10^{-122}$	$1.11 \cdot 10^{-221}$
Waddell – Kappa > Waddell – MI	$1.21 \cdot 10^{-67}$	0.00

A.6.2 Comparing numbers of sequences

In this section, we compare the performance of each method for different numbers of sequences, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

CAPS

Precisions

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
6 > 22	$3.09 \cdot 10^{-101}$
10 > 22	$1.87 \cdot 10^{-120}$

Sensitivities

Numbers of sequences	Wilcoxon p-value
6 > 10	$2.26 \cdot 10^{-18}$
6 > 22	0.00
10 > 22	0.00

Specificities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$5.51 \cdot 10^{-09}$
$6 > 22$	$5.98 \cdot 10^{-16}$
$10 > 22$	$9.49 \cdot 10^{-17}$

PlotCorr

Precisions – no differences

Sensitivities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$1.22 \cdot 10^{-09}$
$6 > 22$	$5.14 \cdot 10^{-171}$
$10 > 22$	$4.43 \cdot 10^{-146}$

Specificities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$3.41 \cdot 10^{-47}$
$6 > 22$	$4.15 \cdot 10^{-196}$
$10 > 22$	$9.23 \cdot 10^{-71}$

MI

Precisions

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	$5.13 \cdot 10^{-116}$
$10 > 22$	$8.09 \cdot 10^{-121}$

Sensitivities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$2.95 \cdot 10^{-10}$
$6 > 22$	0.00
$10 > 22$	0.00

Specificities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$2.35 \cdot 10^{-59}$
$6 > 22$	0.00
$10 > 22$	0.00

Waddell – Kappa

Precisions

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$22 > 6$	$1.25 \cdot 10^{-18}$
$22 > 10$	$2.94 \cdot 10^{-14}$

Sensitivities

Numbers of sequences	Wilcoxon p-value
$10 > 6$	$5.45 \cdot 10^{-05}$
$6 > 22$	0.00
$10 > 22$	$6.29 \cdot 10^{-19}$

Specificities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$9.65 \cdot 10^{-35}$
$6 > 22$	0.01
$22 > 10$	$1.33 \cdot 10^{-36}$

Waddell – MI

Precisions

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	$5.88 \cdot 10^{-07}$
$10 > 22$	$6.70 \cdot 10^{-09}$

Sensitivities

Numbers of sequences	Wilcoxon p-value
$6 > 10$	$1.31 \cdot 10^{-51}$
$6 > 22$	$3.57 \cdot 10^{-138}$
$10 > 22$	$1.08 \cdot 10^{-27}$

Specificities

Numbers of sequences	Wilcoxon p-value
$10 > 6$	0.01
$6 > 22$	0.00
$10 > 22$	0.00

"Higher" UAA threshold

CAPS

Precisions

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	0.00
$10 > 22$	0.00

Sensitivities

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	0.00
$10 > 22$	$2.81 \cdot 10^{-06}$

Specificities – no differences

PlotCorr

Precisions – no differences

Sensitivities

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	0.01
$10 > 22$	0.01

Specificities

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	0.00
$10 > 22$	0.00

MI

Precisions – no differences

Sensitivities – no differences

Specificities

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
$6 > 22$	$2.74 \cdot 10^{-06}$
$10 > 22$	$6.70 \cdot 10^{-12}$

Waddell – Kappa

Precisions – no differences

Sensitivities – no differences

Specificities

Numbers of sequences	Wilcoxon p-value
6 and 10	no difference
6 and 22	no difference
$22 > 10$	0.01

Waddell – MI

Precisions – no differences

Sensitivities – no differences

Specificities

Numbers of sequences	**Wilcoxon p-value**
6 and 10	no difference
$6 > 22$	0.00
10 and 22	no difference

A.7 Results by alignment length p-values

Here we present the p-values of the one-sided Wilcoxon signed-rank and Sign tests for the results when examining the alignment length. If "no difference" is present in one column, then that means that no difference was found between the groups for that test. If "no difference" is present in both columns, then the initial one-sided test was not able to establish a difference between the two groups being compared. The values in the tables are set to show 2 decimal places, so '0.00' is likely to be a number not small enough to be displayed as $\mathrm{X} \cdot 10^{-\mathrm{Y}}$.

A.7.1 Comparing co-evolution detection methods

In this section, we compare the performance of pairs of methods for each alignment length, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

Length 50

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr $>$ CAPS	0.00	0.00
MI $>$ CAPS	$2.63 \cdot 10^{-36}$	$2.53 \cdot 10^{-40}$
Waddell – Kappa $>$ CAPS	0.00	0.00
Waddell – MI $>$ CAPS	$8.58 \cdot 10^{-24}$	$9.69 \cdot 10^{-25}$
PlotCorr $>$ MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa $>$ PlotCorr	$1.46 \cdot 10^{-224}$	$2.86 \cdot 10^{-233}$
PlotCorr $>$ Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa $>$ MI	0.00	0.00
MI $>$ Waddell – MI	$6.04 \cdot 10^{-07}$	$3.56 \cdot 10^{-09}$
Waddell – Kappa $>$ Waddell – MI	0.00	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.81 \cdot 10^{-101}$	$1.73 \cdot 10^{-126}$
CAPS > MI	0.00	0.00
CAPS > Waddell – Kappa	$1.84 \cdot 10^{-06}$	$1.85 \cdot 10^{-07}$
Waddell – MI > CAPS	$1.79 \cdot 10^{-40}$	$2.87 \cdot 10^{-29}$
PlotCorr > MI	$8.38 \cdot 10^{-107}$	$1.11 \cdot 10^{-16}$
PlotCorr > Waddell – Kappa	$2.95 \cdot 10^{-140}$	$2.22 \cdot 10^{-16}$
PlotCorr > Waddell – MI	$8.04 \cdot 10^{-10}$	0.01
MI and Waddell – Kappa	no difference	no difference
Waddell – MI > MI	$7.75 \cdot 10^{-57}$	$8.14 \cdot 10^{-59}$
Waddell – MI > Waddell – Kappa	$9.29 \cdot 10^{-131}$	$9.83 \cdot 10^{-237}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.22 \cdot 10^{-302}$	$8.61 \cdot 10^{-296}$
MI > CAPS	$1.41 \cdot 10^{-34}$	$2.54 \cdot 10^{-66}$
Waddell – Kappa > CAPS	0.00	0.00
CAPS > Waddell – MI	$2.58 \cdot 10^{-05}$	$5.70 \cdot 10^{-05}$
PlotCorr > MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	0.00	0.00
PlotCorr > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	0.00	0.00
MI > Waddell – MI	$6.05 \cdot 10^{-269}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	0.00	$2.22 \cdot 10^{-16}$

Length 600

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$7.33 \cdot 10^{-223}$	0.00
MI > CAPS	$2.75 \cdot 10^{-103}$	$6.76 \cdot 10^{-135}$
Waddell – Kappa > CAPS	$4.06 \cdot 10^{-221}$	0.00
Waddell – MI > CAPS	$4.96 \cdot 10^{-23}$	$3.64 \cdot 10^{-21}$
PlotCorr > MI	$4.24 \cdot 10^{-223}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$4.03 \cdot 10^{-168}$	$3.72 \cdot 10^{-237}$
PlotCorr > Waddell – MI	$2.73 \cdot 10^{-213}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$1.05 \cdot 10^{-220}$	0.00
MI > Waddell – MI	$1.49 \cdot 10^{-17}$	$1.01 \cdot 10^{-12}$
Waddell – Kappa > Waddell – MI	$1.35 \cdot 10^{-221}$	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.60 \cdot 10^{-33}$	$6.97 \cdot 10^{-23}$
MI > CAPS	no difference	0.00
CAPS > Waddell – Kappa	0.00	$1.11 \cdot 10^{-16}$
Waddell – MI > CAPS	$9.95 \cdot 10^{-30}$	$1.27 \cdot 10^{-18}$
PlotCorr > MI	$4.46 \cdot 10^{-47}$	$1.55 \cdot 10^{-15}$
PlotCorr > Waddell – Kappa	$4.72 \cdot 10^{-95}$	0.00
Waddell – MI > PlotCorr	0.02	0.00
MI > Waddell – Kappa	0.01	$2.22 \cdot 10^{-16}$
Waddell – MI > MI	$8.39 \cdot 10^{-45}$	$4.52 \cdot 10^{-27}$
Waddell – MI > Waddell – Kappa	$5.25 \cdot 10^{-112}$	$5.10 \cdot 10^{-203}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$9.27 \cdot 10^{-89}$	$2.09 \cdot 10^{-128}$
MI > CAPS	$2.33 \cdot 10^{-23}$	$1.62 \cdot 10^{-60}$
Waddell – Kappa > CAPS	$9.20 \cdot 10^{-142}$	$2.09 \cdot 10^{-128}$
CAPS > Waddell – MI	no difference	$1.90 \cdot 10^{-06}$
PlotCorr > MI	$4.08 \cdot 10^{-232}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$6.39 \cdot 10^{-232}$	0.00
PlotCorr > Waddell – MI	$2.16 \cdot 10^{-227}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$1.63 \cdot 10^{-232}$	0.00
MI > Waddell – MI	$2.87 \cdot 10^{-130}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$1.63 \cdot 10^{-232}$	$1.11 \cdot 10^{-16}$

Length 1200

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.23 \cdot 10^{-233}$	0.00
MI > CAPS	$2.19 \cdot 10^{-136}$	$1.52 \cdot 10^{-186}$
Waddell – Kappa > CAPS	$3.40 \cdot 10^{-234}$	0.00
Waddell – MI > CAPS	$1.49 \cdot 10^{-50}$	$2.98 \cdot 10^{-26}$
PlotCorr > MI	$2.58 \cdot 10^{-233}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$1.95 \cdot 10^{-204}$	$1.33 \cdot 10^{-286}$
PlotCorr > Waddell – MI	$5.16 \cdot 10^{-234}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$5.41 \cdot 10^{-234}$	0.00
MI > Waddell – MI	$1.01 \cdot 10^{-07}$	$2.36 \cdot 10^{-06}$
Waddell – Kappa > Waddell – MI	$4.01 \cdot 10^{-234}$	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.04 \cdot 10^{-54}$	$4.66 \cdot 10^{-40}$
MI > CAPS	no difference	0.02
CAPS > Waddell – Kappa	no difference	0.02
Waddell – MI > CAPS	$6.75 \cdot 10^{-47}$	$7.24 \cdot 10^{-32}$
PlotCorr > MI	$1.52 \cdot 10^{-84}$	$-2.22 \cdot 10^{-16}$
PlotCorr > Waddell – Kappa	$1.33 \cdot 10^{-63}$	0.00
Waddell – MI > PlotCorr	0.03	$8.22 \cdot 10^{-05}$
Waddell – Kappa > MI	$3.24 \cdot 10^{-16}$	0.00
Waddell – MI > MI	$7.69 \cdot 10^{-76}$	$5.73 \cdot 10^{-70}$
Waddell – MI > Waddell – Kappa	$5.95 \cdot 10^{-91}$	$3.47 \cdot 10^{-164}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.86 \cdot 10^{-82}$	$2.97 \cdot 10^{-84}$
MI > CAPS	$1.80 \cdot 10^{-12}$	$2.03 \cdot 10^{-81}$
Waddell – Kappa > CAPS	$1.32 \cdot 10^{-180}$	$2.97 \cdot 10^{-84}$
CAPS > Waddell – MI	$9.85 \cdot 10^{-07}$	$6.83 \cdot 10^{-11}$
PlotCorr > MI	$2.20 \cdot 10^{-238}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$2.20 \cdot 10^{-238}$	0.00
PlotCorr > Waddell – MI	$2.48 \cdot 10^{-238}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$2.20 \cdot 10^{-238}$	0.00
MI > Waddell – MI	$8.35 \cdot 10^{-207}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$2.20 \cdot 10^{-238}$	$1.11 \cdot 10^{-16}$

"Higher" UAA threshold

Length 50

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$6.59 \cdot 10^{-57}$	$3.00 \cdot 10^{-67}$
MI > CAPS	$3.52 \cdot 10^{-51}$	$3.53 \cdot 10^{-72}$
Waddell – Kappa > CAPS	$5.75 \cdot 10^{-57}$	$1.20 \cdot 10^{-68}$
Waddell – MI > CAPS	$2.07 \cdot 10^{-59}$	$1.70 \cdot 10^{-78}$
PlotCorr > MI	$1.11 \cdot 10^{-56}$	0.00
Waddell – Kappa > PlotCorr	$4.23 \cdot 10^{-44}$	$1.91 \cdot 10^{-57}$
Waddell – MI > PlotCorr	$1.38 \cdot 10^{-38}$	$3.77 \cdot 10^{-56}$
Waddell – Kappa > MI	$1.72 \cdot 10^{-57}$	$1.21 \cdot 10^{-54}$
Waddell – MI > MI	$8.41 \cdot 10^{-59}$	$1.58 \cdot 10^{-71}$
Waddell – Kappa > Waddell – MI	$8.40 \cdot 10^{-21}$	$-2.22 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.31 \cdot 10^{-17}$	$7.50 \cdot 10^{-22}$
MI > CAPS	$3.07 \cdot 10^{-36}$	$3.68 \cdot 10^{-55}$
Waddell – Kappa > CAPS	$3.61 \cdot 10^{-14}$	$4.56 \cdot 10^{-16}$
Waddell – MI > CAPS	$6.06 \cdot 10^{-20}$	$2.84 \cdot 10^{-22}$
MI > PlotCorr	$2.45 \cdot 10^{-17}$	$3.89 \cdot 10^{-25}$
PlotCorr > Waddell – Kappa	0.02	$2.69 \cdot 10^{-05}$
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$2.83 \cdot 10^{-18}$	0.00
MI > Waddell – MI	$7.24 \cdot 10^{-13}$	$1.11 \cdot 10^{-16}$
Waddell – MI > Waddell – Kappa	$4.64 \cdot 10^{-08}$	$2.55 \cdot 10^{-09}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.36 \cdot 10^{-34}$	$2.77 \cdot 10^{-09}$
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	$2.07 \cdot 10^{-37}$	$5.73 \cdot 10^{-11}$
Waddell – MI > CAPS	$1.03 \cdot 10^{-36}$	$5.11 \cdot 10^{-10}$
PlotCorr > MI	$2.32 \cdot 10^{-62}$	0.00
Waddell – Kappa > PlotCorr	$2.32 \cdot 10^{-62}$	$1.66 \cdot 10^{-111}$
Waddell – MI > PlotCorr	$9.87 \cdot 10^{-62}$	$2.57 \cdot 10^{-95}$
Waddell – Kappa > MI	$2.32 \cdot 10^{-62}$	$1.66 \cdot 10^{-111}$
Waddell – MI > MI	$2.32 \cdot 10^{-62}$	$1.66 \cdot 10^{-111}$
Waddell – Kappa > Waddell – MI	$5.66 \cdot 10^{-41}$	0.00

Length 600

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$9.93 \cdot 10^{-32}$	$5.37 \cdot 10^{-51}$
MI > CAPS	$4.51 \cdot 10^{-32}$	$1.50 \cdot 10^{-53}$
Waddell – Kappa > CAPS	$4.44 \cdot 10^{-32}$	$8.16 \cdot 10^{-56}$
Waddell – MI > CAPS	$4.44 \cdot 10^{-32}$	$8.16 \cdot 10^{-56}$
PlotCorr > MI	$1.38 \cdot 10^{-30}$	0.00
Waddell – Kappa > PlotCorr	$4.13 \cdot 10^{-32}$	$3.59 \cdot 10^{-47}$
Waddell – MI > PlotCorr	$4.93 \cdot 10^{-32}$	$1.08 \cdot 10^{-45}$
Waddell – Kappa > MI	$2.30 \cdot 10^{-32}$	$2.15 \cdot 10^{-50}$
Waddell – MI > MI	$2.19 \cdot 10^{-32}$	$3.51 \cdot 10^{-52}$
Waddell – Kappa > Waddell – MI	0.02	$7.12 \cdot 10^{-05}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.49 \cdot 10^{-12}$	$8.24 \cdot 10^{-06}$
MI > CAPS	$5.90 \cdot 10^{-19}$	$2.25 \cdot 10^{-25}$
Waddell – Kappa > CAPS	$9.55 \cdot 10^{-14}$	$1.34 \cdot 10^{-06}$
Waddell – MI > CAPS	$2.98 \cdot 10^{-14}$	$7.41 \cdot 10^{-07}$
MI > PlotCorr	$9.90 \cdot 10^{-13}$	$5.23 \cdot 10^{-21}$
PlotCorr and Waddell – Kappa	no difference	no difference
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$2.36 \cdot 10^{-09}$	$1.44 \cdot 10^{-15}$
MI > Waddell – MI	$2.82 \cdot 10^{-08}$	$6.89 \cdot 10^{-14}$
Waddell – MI > Waddell – Kappa	0.00	$7.25 \cdot 10^{-05}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.03 \cdot 10^{-11}$	0.02
MI > CAPS	no difference	0.02
Waddell – Kappa > CAPS	$1.26 \cdot 10^{-14}$	0.02
Waddell – MI > CAPS	$1.33 \cdot 10^{-14}$	0.02
PlotCorr > MI	$1.71 \cdot 10^{-32}$	0.00
Waddell – Kappa > PlotCorr	$1.43 \cdot 10^{-32}$	$1.02 \cdot 10^{-56}$
Waddell – MI > PlotCorr	$1.43 \cdot 10^{-32}$	$1.02 \cdot 10^{-56}$
Waddell – Kappa > MI	$1.43 \cdot 10^{-32}$	$1.02 \cdot 10^{-56}$
Waddell – MI > MI	$1.43 \cdot 10^{-32}$	$1.02 \cdot 10^{-56}$
Waddell – Kappa > Waddell – MI	$8.03 \cdot 10^{-07}$	$4.69 \cdot 10^{-07}$

Length 1200

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.56 \cdot 10^{-34}$	$4.95 \cdot 10^{-58}$
MI > CAPS	$1.60 \cdot 10^{-35}$	$3.89 \cdot 10^{-62}$
Waddell – Kappa > CAPS	$2.28 \cdot 10^{-34}$	$9.86 \cdot 10^{-58}$
Waddell – MI > CAPS	$3.45 \cdot 10^{-35}$	$3.16 \cdot 10^{-59}$
PlotCorr > MI	$2.97 \cdot 10^{-35}$	0.00
Waddell – Kappa > PlotCorr	$1.27 \cdot 10^{-34}$	$8.30 \cdot 10^{-52}$
Waddell – MI > PlotCorr	$5.42 \cdot 10^{-35}$	$1.08 \cdot 10^{-55}$
Waddell – Kappa > MI	$2.12 \cdot 10^{-35}$	$3.00 \cdot 10^{-47}$
Waddell – MI > MI	$1.27 \cdot 10^{-35}$	$1.42 \cdot 10^{-54}$
Waddell – Kappa > Waddell – MI	$1.27 \cdot 10^{-13}$	$1.25 \cdot 10^{-12}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$5.70 \cdot 10^{-23}$	$1.02 \cdot 10^{-17}$
MI > CAPS	$2.54 \cdot 10^{-28}$	$2.01 \cdot 10^{-40}$
Waddell – Kappa > CAPS	$3.68 \cdot 10^{-23}$	$1.99 \cdot 10^{-18}$
Waddell – MI > CAPS	$1.67 \cdot 10^{-24}$	$8.46 \cdot 10^{-20}$
MI > PlotCorr	$5.73 \cdot 10^{-17}$	$7.81 \cdot 10^{-27}$
PlotCorr and Waddell – Kappa	no difference	no difference
Waddell – MI > PlotCorr	0.02	0.01
MI > Waddell – Kappa	$1.51 \cdot 10^{-12}$	0.00
MI > Waddell – MI	$1.08 \cdot 10^{-09}$	$1.97 \cdot 10^{-14}$
Waddell – MI > Waddell – Kappa	$1.94 \cdot 10^{-06}$	$2.31 \cdot 10^{-07}$

Specificities

Methods	Wilcoxon p-value	Sign p-value
CAPS > PlotCorr	no difference	0.00
CAPS > MI	0.01	0.00
CAPS > Waddell – Kappa	no difference	0.00
CAPS > Waddell – MI	no difference	0.00
PlotCorr > MI	$1.10 \cdot 10^{-35}$	0.00
Waddell – Kappa > PlotCorr	$1.10 \cdot 10^{-35}$	$1.94 \cdot 10^{-62}$
Waddell – MI > PlotCorr	$1.10 \cdot 10^{-35}$	$1.94 \cdot 10^{-62}$
Waddell – Kappa > MI	$1.10 \cdot 10^{-35}$	$1.94 \cdot 10^{-62}$
Waddell – MI > MI	$1.10 \cdot 10^{-35}$	$1.94 \cdot 10^{-62}$
Waddell – Kappa > Waddell – MI	$2.49 \cdot 10^{-25}$	0.00

A.7.2 Comparing alignment lengths

In this section, we compare the performance of each method for different alignment lengths, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

CAPS

Precisions

Methods	Wilcoxon p-value
50 > 600	$5.98 \cdot 10^{-20}$
50 > 1200	$5.76 \cdot 10^{-165}$
600 > 1200	$5.70 \cdot 10^{-48}$

Sensitivities

Methods	Wilcoxon p-value
50 and 600	no difference
50 and 1200	no difference
600 > 1200	0.02

Specificities

Methods	Wilcoxon p-value
$600 > 50$	$2.66 \cdot 10^{-42}$
$1200 > 50$	$5.81 \cdot 10^{-19}$
$600 > 1200$	0.00

PlotCorr

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$1.09 \cdot 10^{-60}$
$50 > 1200$	$1.07 \cdot 10^{-270}$
$600 > 1200$	$1.55 \cdot 10^{-63}$

Sensitivities

Methods	Wilcoxon p-value
$50 > 600$	0.01
$50 > 1200$	0.01
600 and 1200	no difference

Specificities

Methods	Wilcoxon p-value
$600 > 50$	$5.81 \cdot 10^{-11}$
$50 > 1200$	$8.46 \cdot 10^{-10}$
$600 > 1200$	$1.11 \cdot 10^{-23}$

MI

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$8.66 \cdot 10^{-08}$
$50 > 1200$	$2.66 \cdot 10^{-108}$
$600 > 1200$	$2.16 \cdot 10^{-53}$

Sensitivities

Methods	Wilcoxon p-value
$600 > 50$	$2.57 \cdot 10^{-07}$
50 and 1200	no difference
$600 > 1200$	0.00

Specificities

Methods	Wilcoxon p-value
$600 > 50$	$1.14 \cdot 10^{-71}$
$1200 > 50$	$3.53 \cdot 10^{-20}$
$600 > 1200$	$2.24 \cdot 10^{-21}$

Waddell – Kappa

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$1.77 \cdot 10^{-43}$
$50 > 1200$	$6.96 \cdot 10^{-204}$
$600 > 1200$	$9.66 \cdot 10^{-59}$

Sensitivities

Methods	Wilcoxon p-value
$600 > 50$	$4.11 \cdot 10^{-06}$
$1200 > 50$	$8.93 \cdot 10^{-16}$
$1200 > 600$	$3.89 \cdot 10^{-06}$

Specificities

Methods	Wilcoxon p-value
$600 > 50$	0.01
$50 > 1200$	$1.21 \cdot 10^{-26}$
$600 > 1200$	$1.61 \cdot 10^{-26}$

Waddell – MI

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$7.82 \cdot 10^{-43}$
$50 > 1200$	$1.72 \cdot 10^{-263}$
$600 > 1200$	$2.72 \cdot 10^{-67}$

Sensitivities

Methods	Wilcoxon p-value
$600 > 50$	$6.49 \cdot 10^{-09}$
$1200 > 50$	$1.06 \cdot 10^{-06}$
600 and 1200	no difference

Specificities

Methods	Wilcoxon p-value
$600 > 50$	$7.66 \cdot 10^{-76}$
$1200 > 50$	$3.60 \cdot 10^{-13}$
$600 > 1200$	$3.48 \cdot 10^{-32}$

"Upper" UAA threshold

CAPS

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$7.20 \cdot 10^{-06}$
$50 > 1200$	$4.51 \cdot 10^{-16}$
$600 > 1200$	$1.01 \cdot 10^{-08}$

Sensitivities

Methods	Wilcoxon p-value
50 and 600	no difference
$50 > 1200$	0.00
$600 > 1200$	0.00

Specificities

Methods	Wilcoxon p-value
50 and 600	no difference
1200 $>$ 50	0.00
1200 $>$ 600	0.01

PlotCorr

Precisions

Methods	Wilcoxon p-value
50 $>$ 600	$2.74 \cdot 10^{-32}$
50 $>$ 1200	$2.57 \cdot 10^{-47}$
600 $>$ 1200	$2.25 \cdot 10^{-11}$

Sensitivities – no differences

Specificities

Methods	Wilcoxon p-value
50 $>$ 600	$5.53 \cdot 10^{-12}$
50 $>$ 1200	$1.66 \cdot 10^{-12}$
600 and 1200	no difference

MI

Precisions

Methods	Wilcoxon p-value
50 $>$ 600	$7.42 \cdot 10^{-41}$
50 $>$ 1200	$2.23 \cdot 10^{-73}$
600 $>$ 1200	$1.29 \cdot 10^{-12}$

Sensitivities – no differences

Specificities

Methods	Wilcoxon p-value
600 $>$ 50	$6.38 \cdot 10^{-11}$
1200 $>$ 50	0.00
600 $>$ 1200	0.00

Waddell – Kappa

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$1.00 \cdot 10^{-37}$
$50 > 1200$	$4.54 \cdot 10^{-50}$
$600 > 1200$	$3.61 \cdot 10^{-11}$

Sensitivities

Methods	Wilcoxon p-value
$600 > 50$	0.00
50 and 1200	no difference
600 and 1200	no difference

Specificities

Methods	Wilcoxon p-value
$50 > 600$	$9.79 \cdot 10^{-29}$
$50 > 1200$	$7.90 \cdot 10^{-22}$
$1200 > 600$	0.01

Waddell – MI

Precisions

Methods	Wilcoxon p-value
$50 > 600$	$7.34 \cdot 10^{-41}$
$50 > 1200$	$1.99 \cdot 10^{-62}$
$600 > 1200$	$6.17 \cdot 10^{-12}$

Sensitivities – no differences

Specificities

Methods	Wilcoxon p-value
$50 > 600$	$3.65 \cdot 10^{-15}$
$50 > 1200$	$2.83 \cdot 10^{-24}$
$600 > 1200$	0.01

A.8 Results by rate variation parameter p-values

Here we present the p-values of the one-sided Wilcoxon signed-rank and Sign tests for the results when examining the rate variation parameter. If "no difference" is present in one column, then that means that no difference was found between the groups for that test. If "no difference" is present in both columns, then the initial one-sided test was not able to establish a difference between the two groups being compared. The values in the tables are set to show 2 decimal places, so '0.00' is likely to be a number not small enough to be displayed as $\mathrm{X} \cdot 10^{-\mathrm{Y}}$.

A.8.1 Comparing co-evolution detection methods

In this section, we compare the performance of pairs of methods for each rate variation parameter value, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

Value 1

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$4.28 \cdot 10^{-259}$	0.00
MI > CAPS	$9.16 \cdot 10^{-37}$	$1.14 \cdot 10^{-50}$
Waddell – Kappa > CAPS	$9.04 \cdot 10^{-234}$	0.00
Waddell – MI > CAPS	$5.42 \cdot 10^{-08}$	$9.48 \cdot 10^{-09}$
PlotCorr > MI	$2.90 \cdot 10^{-265}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$2.87 \cdot 10^{-156}$	$4.87 \cdot 10^{-228}$
PlotCorr > Waddell – MI	$3.08 \cdot 10^{-257}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$4.63 \cdot 10^{-229}$	0.00
MI > Waddell – MI	$5.06 \cdot 10^{-21}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$3.18 \cdot 10^{-241}$	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.44 \cdot 10^{-08}$	$4.38 \cdot 10^{-08}$
CAPS > MI	$2.67 \cdot 10^{-50}$	$-2.22 \cdot 10^{-16}$
CAPS > Waddell – Kappa	$8.22 \cdot 10^{-58}$	$1.11 \cdot 10^{-16}$
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$2.80 \cdot 10^{-74}$	$-2.22 \cdot 10^{-16}$
PlotCorr > Waddell – Kappa	$2.12 \cdot 10^{-97}$	0.00
PlotCorr > Waddell – MI	$2.40 \cdot 10^{-07}$	0.01
MI > Waddell – Kappa	no difference	0.01
Waddell – MI > MI	$1.92 \cdot 10^{-44}$	$2.41 \cdot 10^{-38}$
Waddell – MI > Waddell – Kappa	$8.18 \cdot 10^{-93}$	$1.70 \cdot 10^{-167}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$4.11 \cdot 10^{-254}$	0.00
MI > CAPS	$4.88 \cdot 10^{-194}$	$3.61 \cdot 10^{-215}$
Waddell – Kappa > CAPS	$4.33 \cdot 10^{-273}$	0.00
Waddell – MI > CAPS	$2.47 \cdot 10^{-64}$	$7.30 \cdot 10^{-14}$
PlotCorr > MI	$2.25 \cdot 10^{-279}$	0.00
Waddell – Kappa > PlotCorr	$1.01 \cdot 10^{-278}$	0.00
PlotCorr > Waddell – MI	$3.21 \cdot 10^{-276}$	0.00
Waddell – Kappa > MI	$6.00 \cdot 10^{-280}$	0.00
MI > Waddell – MI	$1.12 \cdot 10^{-202}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$6.00 \cdot 10^{-280}$	$1.11 \cdot 10^{-16}$

Value 100

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.16 \cdot 10^{-275}$	0.00
MI > CAPS	$2.15 \cdot 10^{-52}$	$7.18 \cdot 10^{-95}$
Waddell – Kappa > CAPS	$2.46 \cdot 10^{-249}$	0.00
Waddell – MI > CAPS	$1.48 \cdot 10^{-31}$	$7.47 \cdot 10^{-31}$
PlotCorr > MI	$8.69 \cdot 10^{-280}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$9.84 \cdot 10^{-170}$	$2.11 \cdot 10^{-243}$
PlotCorr > Waddell – MI	$6.63 \cdot 10^{-263}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$1.15 \cdot 10^{-244}$	0.00
MI > Waddell – MI	0.00	0.02
Waddell – Kappa > Waddell – MI	$1.20 \cdot 10^{-251}$	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.36 \cdot 10^{-91}$	$1.53 \cdot 10^{-87}$
MI > CAPS	no difference	0.00
CAPS and Waddell – Kappa	no difference	no difference
Waddell – MI > CAPS	$2.25 \cdot 10^{-61}$	$4.80 \cdot 10^{-41}$
PlotCorr > MI	$4.60 \cdot 10^{-94}$	0.00
PlotCorr > Waddell – Kappa	$4.50 \cdot 10^{-94}$	0.00
PlotCorr and Waddell – MI	no difference	no difference
MI and Waddell – Kappa	no difference	no difference
Waddell – MI > MI	$2.66 \cdot 10^{-67}$	$4.01 \cdot 10^{-58}$
Waddell – MI > Waddell – Kappa	$1.93 \cdot 10^{-102}$	$2.94 \cdot 10^{-185}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$6.19 \cdot 10^{-118}$	$9.24 \cdot 10^{-106}$
MI > CAPS	0.04	$2.81 \cdot 10^{-33}$
Waddell – Kappa > CAPS	$2.68 \cdot 10^{-204}$	$4.96 \cdot 10^{-115}$
CAPS > Waddell – MI	$3.25 \cdot 10^{-21}$	0.00
PlotCorr > MI	$3.97 \cdot 10^{-294}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$7.88 \cdot 10^{-293}$	0.00
PlotCorr > Waddell – MI	$1.84 \cdot 10^{-287}$	0.00
Waddell – Kappa > MI	$6.07 \cdot 10^{-295}$	0.00
MI > Waddell – MI	$8.24 \cdot 10^{-191}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	$1.87 \cdot 10^{-294}$	$1.11 \cdot 10^{-16}$

Value none

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.68 \cdot 10^{-306}$	0.00
MI > CAPS	$2.65 \cdot 10^{-94}$	$4.11 \cdot 10^{-156}$
Waddell – Kappa > CAPS	$4.62 \cdot 10^{-293}$	0.00
Waddell – MI > CAPS	$1.51 \cdot 10^{-43}$	$2.24 \cdot 10^{-36}$
PlotCorr > MI	$3.41 \cdot 10^{-306}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$2.85 \cdot 10^{-189}$	$1.21 \cdot 10^{-251}$
PlotCorr > Waddell – MI	$1.34 \cdot 10^{-281}$	$2.22 \cdot 10^{-16}$
Waddell – Kappa > MI	$1.46 \cdot 10^{-281}$	0.00
MI > Waddell – MI	$1.14 \cdot 10^{-07}$	$2.65 \cdot 10^{-07}$
Waddell – Kappa > Waddell – MI	$2.09 \cdot 10^{-293}$	$2.22 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$4.26 \cdot 10^{-106}$	$2.19 \cdot 10^{-100}$
MI > CAPS	$2.25 \cdot 10^{-17}$	$5.48 \cdot 10^{-21}$
CAPS and Waddell – Kappa	no difference	no difference
Waddell – MI > CAPS	$3.71 \cdot 10^{-82}$	$2.34 \cdot 10^{-61}$
PlotCorr > MI	$6.54 \cdot 10^{-73}$	0.00
PlotCorr > Waddell – Kappa	$7.74 \cdot 10^{-107}$	$1.11 \cdot 10^{-16}$
Waddell – MI > PlotCorr	0.03	$6.30 \cdot 10^{-09}$
MI > Waddell – Kappa	no difference	$3.80 \cdot 10^{-05}$
Waddell – MI > MI	$6.31 \cdot 10^{-58}$	$3.26 \cdot 10^{-53}$
Waddell – MI > Waddell – Kappa	$3.70 \cdot 10^{-138}$	$3.49 \cdot 10^{-251}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.79 \cdot 10^{-120}$	$1.26 \cdot 10^{-96}$
MI > CAPS	no difference	$9.95 \cdot 10^{-26}$
Waddell – Kappa > CAPS	$3.63 \cdot 10^{-221}$	$8.57 \cdot 10^{-108}$
CAPS > Waddell – MI	$7.61 \cdot 10^{-40}$	$1.11 \cdot 10^{-16}$
PlotCorr > MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	0.00	0.00
PlotCorr > Waddell – MI	0.00	$2.22 \cdot 10^{-16}$
Waddell – Kappa > MI	0.00	0.00
MI > Waddell – MI	$5.75 \cdot 10^{-212}$	$2.22 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$

"Higher" UAA threshold

Value 1

Precisions

Methods	Wilcoxon p-value	Sign p-value
PlotCorr > CAPS	$8.18 \cdot 10^{-32}$	$6.93 \cdot 10^{-45}$
MI > CAPS	$1.76 \cdot 10^{-36}$	$3.04 \cdot 10^{-60}$
Waddell – Kappa > CAPS	$4.69 \cdot 10^{-36}$	$1.33 \cdot 10^{-54}$
Waddell – MI > CAPS	$5.11 \cdot 10^{-36}$	$1.33 \cdot 10^{-54}$
PlotCorr > MI	$3.65 \cdot 10^{-29}$	0.00
Waddell – Kappa > PlotCorr	$1.05 \cdot 10^{-33}$	$5.21 \cdot 10^{-56}$
Waddell – MI > PlotCorr	$1.62 \cdot 10^{-30}$	$1.50 \cdot 10^{-50}$
Waddell – Kappa > MI	$5.01 \cdot 10^{-36}$	$1.03 \cdot 10^{-55}$
Waddell – MI > MI	$6.35 \cdot 10^{-36}$	$1.03 \cdot 10^{-55}$
Waddell – Kappa > Waddell – MI	$2.77 \cdot 10^{-07}$	$1.22 \cdot 10^{-08}$

Sensitivities

Methods	Wilcoxon p-value	Sign p-value
CAPS and PlotCorr	no difference	no difference
MI > CAPS	$3.96 \cdot 10^{-09}$	$1.58 \cdot 10^{-12}$
CAPS and Waddell – Kappa	no difference	no difference
CAPS and Waddell – MI	no difference	no difference
MI > PlotCorr	$1.07 \cdot 10^{-13}$	$2.61 \cdot 10^{-22}$
PlotCorr and Waddell – Kappa	no difference	no difference
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$1.19 \cdot 10^{-10}$	0.00
MI > Waddell – MI	$1.73 \cdot 10^{-10}$	$2.22 \cdot 10^{-16}$
Waddell – Kappa and Waddell – MI	no difference	no difference

Specificities

Methods	Wilcoxon p-value	Sign p-value
PlotCorr > CAPS	$4.05 \cdot 10^{-38}$	$5.78 \cdot 10^{-46}$
MI > CAPS	$2.59 \cdot 10^{-26}$	$1.02 \cdot 10^{-39}$
Waddell – Kappa > CAPS	$3.56 \cdot 10^{-39}$	$5.78 \cdot 10^{-46}$
Waddell – MI > CAPS	$3.56 \cdot 10^{-39}$	$5.78 \cdot 10^{-46}$
PlotCorr > MI	$6.85 \cdot 10^{-40}$	0.00
Waddell – Kappa > PlotCorr	$6.02 \cdot 10^{-40}$	$2.90 \cdot 10^{-70}$
Waddell – MI > PlotCorr	$6.02 \cdot 10^{-40}$	$2.90 \cdot 10^{-70}$
Waddell – Kappa > MI	$6.02 \cdot 10^{-40}$	$2.90 \cdot 10^{-70}$
Waddell – MI > MI	$6.02 \cdot 10^{-40}$	$2.90 \cdot 10^{-70}$
Waddell – Kappa > Waddell – MI	$6.96 \cdot 10^{-09}$	$3.06 \cdot 10^{-08}$

Value 100

Precisions

Methods	Wilcoxon p-value	Sign p-value
PlotCorr > CAPS	$5.25 \cdot 10^{-32}$	$1.58 \cdot 10^{-52}$
MI > CAPS	$1.59 \cdot 10^{-27}$	$2.10 \cdot 10^{-46}$
Waddell – Kappa > CAPS	$5.35 \cdot 10^{-30}$	$2.04 \cdot 10^{-47}$
Waddell – MI > CAPS	$6.80 \cdot 10^{-32}$	$4.26 \cdot 10^{-52}$
PlotCorr > MI	$5.49 \cdot 10^{-32}$	0.00
Waddell – Kappa > PlotCorr	$4.35 \cdot 10^{-24}$	$9.90 \cdot 10^{-39}$
Waddell – MI > PlotCorr	$1.03 \cdot 10^{-22}$	$2.29 \cdot 10^{-40}$
Waddell – Kappa > MI	$2.16 \cdot 10^{-30}$	$6.59 \cdot 10^{-41}$
Waddell – MI > MI	$5.60 \cdot 10^{-32}$	$1.06 \cdot 10^{-49}$
Waddell – Kappa > Waddell – MI	$2.62 \cdot 10^{-15}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.03 \cdot 10^{-21}$	$1.97 \cdot 10^{-23}$
MI > CAPS	$3.95 \cdot 10^{-28}$	$5.09 \cdot 10^{-41}$
Waddell – Kappa > CAPS	$4.40 \cdot 10^{-20}$	$3.86 \cdot 10^{-20}$
Waddell – MI > CAPS	$1.07 \cdot 10^{-22}$	$2.73 \cdot 10^{-23}$
MI > PlotCorr	$5.72 \cdot 10^{-11}$	$1.32 \cdot 10^{-17}$
PlotCorr and Waddell – Kappa	no difference	no difference
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$6.70 \cdot 10^{-09}$	$2.44 \cdot 10^{-13}$
MI > Waddell – MI	$1.92 \cdot 10^{-06}$	$6.42 \cdot 10^{-10}$
Waddell – MI > Waddell – Kappa	$4.95 \cdot 10^{-05}$	$2.00 \cdot 10^{-05}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
CAPS and PlotCorr	no difference	no difference
CAPS > MI	$2.61 \cdot 10^{-10}$	0.01
CAPS and Waddell – Kappa	no difference	no difference
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$5.15 \cdot 10^{-36}$	0.00
Waddell – Kappa > PlotCorr	$5.15 \cdot 10^{-36}$	$4.86 \cdot 10^{-63}$
Waddell – MI > PlotCorr	$1.01 \cdot 10^{-35}$	$1.42 \cdot 10^{-56}$
Waddell – Kappa > MI	$5.15 \cdot 10^{-36}$	$4.86 \cdot 10^{-63}$
Waddell – MI > MI	$5.15 \cdot 10^{-36}$	$4.86 \cdot 10^{-63}$
Waddell – Kappa > Waddell – MI	$7.40 \cdot 10^{-26}$	0.00

Value none

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$6.50 \cdot 10^{-47}$	$1.52 \cdot 10^{-76}$
MI > CAPS	$7.09 \cdot 10^{-47}$	$2.58 \cdot 10^{-76}$
Waddell – Kappa > CAPS	$4.99 \cdot 10^{-47}$	$5.51 \cdot 10^{-75}$
Waddell – MI > CAPS	$1.37 \cdot 10^{-50}$	$1.13 \cdot 10^{-83}$
PlotCorr > MI	$2.37 \cdot 10^{-39}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$3.11 \cdot 10^{-40}$	$3.07 \cdot 10^{-59}$
Waddell – MI > PlotCorr	$7.75 \cdot 10^{-42}$	$5.65 \cdot 10^{-62}$
Waddell – Kappa > MI	$2.60 \cdot 10^{-41}$	$1.06 \cdot 10^{-52}$
Waddell – MI > MI	$1.21 \cdot 10^{-47}$	$3.70 \cdot 10^{-70}$
Waddell – Kappa > Waddell – MI	$5.06 \cdot 10^{-11}$	$1.39 \cdot 10^{-12}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.52 \cdot 10^{-33}$	$1.16 \cdot 10^{-38}$
MI > CAPS	$1.67 \cdot 10^{-46}$	$8.25 \cdot 10^{-69}$
Waddell – Kappa > CAPS	$1.11 \cdot 10^{-31}$	$2.93 \cdot 10^{-34}$
Waddell – MI > CAPS	$3.15 \cdot 10^{-38}$	$5.70 \cdot 10^{-42}$
MI > PlotCorr	$5.76 \cdot 10^{-21}$	$9.30 \cdot 10^{-33}$
PlotCorr > Waddell – Kappa	0.01	0.01
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$1.42 \cdot 10^{-20}$	0.00
MI > Waddell – MI	$1.42 \cdot 10^{-13}$	0.00
Waddell – MI > Waddell – Kappa	$7.45 \cdot 10^{-13}$	$1.36 \cdot 10^{-20}$

Specificities

Methods	Wilcoxon p-value	Sign p-value
CAPS > PlotCorr	no difference	$3.87 \cdot 10^{-07}$
CAPS > MI	$1.56 \cdot 10^{-23}$	$7.67 \cdot 10^{-14}$
CAPS > Waddell – Kappa	no difference	$7.18 \cdot 10^{-06}$
CAPS > Waddell – MI	no difference	$1.54 \cdot 10^{-06}$
PlotCorr > MI	$1.12 \cdot 10^{-54}$	0.00
Waddell – Kappa > PlotCorr	$1.12 \cdot 10^{-54}$	$2.34 \cdot 10^{-97}$
Waddell – MI > PlotCorr	$1.83 \cdot 10^{-54}$	$6.55 \cdot 10^{-87}$
Waddell – Kappa > MI	$1.12 \cdot 10^{-54}$	$2.34 \cdot 10^{-97}$
Waddell – MI > MI	$1.12 \cdot 10^{-54}$	$2.34 \cdot 10^{-97}$
Waddell – Kappa > Waddell – MI	$3.23 \cdot 10^{-36}$	0.00

A.8.2 Comparing rate variation parameter values

In this section, we compare the performance of each method for different rate variation parameter values, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

CAPS

Precisions

Methods	Wilcoxon p-value
1 > none	$3.69 \cdot 10^{-22}$
none and 100	no difference
1 > 100	$2.87 \cdot 10^{-23}$

Sensitivities

Methods	Wilcoxon p-value
1 > none	$2.85 \cdot 10^{-46}$
100 > none	0.02
1 > 100	$1.29 \cdot 10^{-31}$

Specificities

Methods	Wilcoxon p-value
none $>$ 1	$6.76 \cdot 10^{-59}$
none and 100	no difference
100 $>$ 1	$5.73 \cdot 10^{-45}$

PlotCorr

Precisions – no differences

Sensitivities

Methods	Wilcoxon p-value
1 $>$ none	$6.35 \cdot 10^{-09}$
100 $>$ none	$2.30 \cdot 10^{-05}$
1 and 100	no difference

Specificities

Methods	Wilcoxon p-value
none $>$ 1	$4.77 \cdot 10^{-06}$
none and 100	no difference
100 $>$ 1	0.00

MI

Precisions

Methods	Wilcoxon p-value
none and 1	no difference
none $>$ 100	0.00
1 $>$ 100	$8.19 \cdot 10^{-07}$

Sensitivities

Methods	Wilcoxon p-value
none and 1	no difference
none and 100	no difference
1 $>$ 100	0.00

Specificities

Methods	Wilcoxon p-value
$1 >$ none	$1.23 \cdot 10^{-23}$
none and 100	no difference
$1 > 100$	$3.21 \cdot 10^{-23}$

Waddell – Kappa

Precisions

Methods	Wilcoxon p-value
none and 1	no difference
none > 100	0.02
1 and 100	no difference

Sensitivities

Methods	Wilcoxon p-value
$1 >$ none	0.00
$100 >$ none	$9.46 \cdot 10^{-05}$
1 and 100	no difference

Specificities

Methods	Wilcoxon p-value
none > 1	$3.54 \cdot 10^{-07}$
none and 100	no difference
$100 > 1$	$2.99 \cdot 10^{-06}$

Waddell – MI

Precisions

Methods	Wilcoxon p-value
none and 1	no difference
none > 100	0.00
$1 > 100$	0.00

Sensitivities – no differences

Specificities

Methods	Wilcoxon p-value
$1 >$ none	$2.79 \cdot 10^{-29}$
none and 100	no difference
$1 > 100$	$1.39 \cdot 10^{-25}$

"Upper" UAA threshold

CAPS

Precisions

Methods	Wilcoxon p-value
$1 >$ none	$3.40 \cdot 10^{-28}$
$100 >$ none	0.00
$1 > 100$	$7.42 \cdot 10^{-13}$

Sensitivities

Methods	Wilcoxon p-value
$1 >$ none	$4.31 \cdot 10^{-42}$
$100 >$ none	0.00
$1 > 100$	$4.52 \cdot 10^{-26}$

Specificities

Methods	Wilcoxon p-value
none > 1	$8.49 \cdot 10^{-45}$
none > 100	0.01
$100 > 1$	$1.65 \cdot 10^{-27}$

PlotCorr

Precisions – no differences

Sensitivities

Methods	Wilcoxon p-value
$1 >$ none	0.00
$100 >$ none	0.00
1 and 100	no difference

Specificities

Methods	Wilcoxon p-value
none > 1	$2.58 \cdot 10^{-27}$
none > 100	0.00
$100 > 1$	$8.20 \cdot 10^{-17}$

MI

Precisions

Methods	Wilcoxon p-value
none and 1	no difference
none > 100	0.01
$1 > 100$	0.01

Sensitivities

Methods	Wilcoxon p-value
$1 >$ none	0.00
$100 >$ none	0.01
1 and 100	no difference

Specificities

Methods	Wilcoxon p-value
$1 >$ none	$9.09 \cdot 10^{-17}$
none and 100	no difference
$1 > 100$	$4.41 \cdot 10^{-11}$

Waddell – Kappa

Precisions – no differences

Sensitivities

Methods	Wilcoxon p-value
$1 >$ none	$3.13 \cdot 10^{-07}$
$100 >$ none	$4.09 \cdot 10^{-05}$
1 and 100	no difference

Specificities

Methods	Wilcoxon p-value
none > 1	$7.89 \cdot 10^{-22}$
none > 100	0.00
$100 > 1$	$7.31 \cdot 10^{-13}$

Waddell – MI

Precisions

Methods	Wilcoxon p-value
none and 1	no difference
none > 100	0.00
1 and 100	no difference

Sensitivities

Methods	Wilcoxon p-value
$1 >$ none	0.02
$100 >$ none	0.00
1 and 100	no difference

Specificities

Methods	Wilcoxon p-value
none > 1	0.00
none > 100	0.00
1 and 100	no difference

A.9 Results by percentage of sites co-evolving p-values

Here we present the p-values of the one-sided Wilcoxon signed-rank and Sign tests for the results when examining the percentage of sites co-evolving. If "no difference" is present in one column, then that means that no difference was found between the groups for that test. If "no difference" is present in both columns, then the initial one-sided test was not able to establish a difference between the two groups being compared. The values in the tables are set to show 2 decimal places, so '0.00' is likely to be a number not small enough to be displayed as $\mathrm{X} \cdot 10^{-\mathrm{Y}}$.

A.9.1 Comparing co-evolution detection methods

In this section, we compare the performance of pairs of methods for each percentage of sites co-evolving, for the "lower" and "higher" unique amino acid thresholds.

"Lower" UAA threshold

0–5%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	0.00	0.00
MI > CAPS	$6.81 \cdot 10^{-95}$	$4.85 \cdot 10^{-174}$
Waddell – Kappa > CAPS	0.00	0.00
Waddell – MI > CAPS	$2.09 \cdot 10^{-05}$	$1.28 \cdot 10^{-08}$
PlotCorr > MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$3.77 \cdot 10^{-244}$	0.00
PlotCorr > Waddell – MI	0.00	$2.22 \cdot 10^{-16}$
Waddell – Kappa > MI	0.00	0.00
MI > Waddell – MI	$1.13 \cdot 10^{-70}$	$2.22 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.37 \cdot 10^{-83}$	$5.35 \cdot 10^{-62}$
CAPS and MI	no difference	no difference
CAPS > Waddell – Kappa	no difference	0.00
Waddell – MI > CAPS	$4.76 \cdot 10^{-80}$	$3.75 \cdot 10^{-61}$
PlotCorr > MI	$9.96 \cdot 10^{-87}$	0.00
PlotCorr > Waddell – Kappa	$4.39 \cdot 10^{-92}$	$1.11 \cdot 10^{-16}$
Waddell – MI > PlotCorr	$4.11 \cdot 10^{-05}$	$9.12 \cdot 10^{-07}$
MI and Waddell – Kappa	no difference	no difference
Waddell – MI > MI	$2.95 \cdot 10^{-87}$	$3.71 \cdot 10^{-87}$
Waddell – MI > Waddell – Kappa	$5.83 \cdot 10^{-128}$	$5.15 \cdot 10^{-231}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.69 \cdot 10^{-175}$	$8.53 \cdot 10^{-206}$
MI > CAPS	$8.50 \cdot 10^{-17}$	$1.63 \cdot 10^{-72}$
Waddell – Kappa > CAPS	0.00	$1.95 \cdot 10^{-220}$
CAPS > Waddell – MI	$2.13 \cdot 10^{-31}$	$1.11 \cdot 10^{-16}$
PlotCorr > MI	0.00	$2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	0.00	0.00
PlotCorr > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	0.00	0.00
MI > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$
Waddell – Kappa > Waddell – MI	0.00	$1.11 \cdot 10^{-16}$

5–10%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.04 \cdot 10^{-134}$	$1.33 \cdot 10^{-225}$
MI > CAPS	$6.10 \cdot 10^{-40}$	$3.81 \cdot 10^{-49}$
Waddell – Kappa > CAPS	$5.81 \cdot 10^{-132}$	$2.97 \cdot 10^{-198}$
Waddell – MI > CAPS	0.01	$2.07 \cdot 10^{-05}$
PlotCorr > MI	$2.44 \cdot 10^{-135}$	0.00
Waddell – Kappa > PlotCorr	$2.63 \cdot 10^{-91}$	$6.97 \cdot 10^{-115}$
PlotCorr > Waddell – MI	$3.36 \cdot 10^{-129}$	0.00
Waddell – Kappa > MI	$1.35 \cdot 10^{-129}$	$8.65 \cdot 10^{-195}$
MI > Waddell – MI	$1.44 \cdot 10^{-26}$	$5.26 \cdot 10^{-10}$
Waddell – Kappa > Waddell – MI	$2.78 \cdot 10^{-134}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.29 \cdot 10^{-24}$	$1.31 \cdot 10^{-17}$
MI > CAPS	no difference	0.02
CAPS > Waddell – Kappa	no difference	$1.31 \cdot 10^{-10}$
Waddell – MI > CAPS	$3.10 \cdot 10^{-19}$	$1.29 \cdot 10^{-16}$
PlotCorr > MI	$2.01 \cdot 10^{-38}$	0.00
PlotCorr > Waddell – Kappa	$1.15 \cdot 10^{-51}$	$-2.22 \cdot 10^{-16}$
Waddell – MI > PlotCorr	no difference	$6.75 \cdot 10^{-06}$
MI > Waddell – Kappa	no difference	0.00
Waddell – MI > MI	$1.39 \cdot 10^{-28}$	$4.89 \cdot 10^{-27}$
Waddell – MI > Waddell – Kappa	$3.47 \cdot 10^{-61}$	$4.26 \cdot 10^{-109}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.23 \cdot 10^{-65}$	$5.15 \cdot 10^{-96}$
MI > CAPS	$6.62 \cdot 10^{-22}$	$1.48 \cdot 10^{-38}$
Waddell – Kappa > CAPS	$4.94 \cdot 10^{-111}$	$1.88 \cdot 10^{-99}$
CAPS > Waddell – MI	no difference	$5.13 \cdot 10^{-05}$
PlotCorr > MI	$8.45 \cdot 10^{-137}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$1.05 \cdot 10^{-136}$	$4.91 \cdot 10^{-235}$
PlotCorr > Waddell – MI	$5.80 \cdot 10^{-132}$	0.00
Waddell – Kappa > MI	$1.88 \cdot 10^{-137}$	$5.59 \cdot 10^{-250}$
MI > Waddell – MI	$1.06 \cdot 10^{-88}$	0.00
Waddell – Kappa > Waddell – MI	$3.99 \cdot 10^{-137}$	0.00

10–15%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.66 \cdot 10^{-88}$	$3.10 \cdot 10^{-138}$
MI > CAPS	$6.28 \cdot 10^{-25}$	$1.36 \cdot 10^{-26}$
Waddell – Kappa > CAPS	$2.21 \cdot 10^{-87}$	$4.82 \cdot 10^{-130}$
Waddell – MI > CAPS	0.00	$7.55 \cdot 10^{-05}$
PlotCorr > MI	$6.24 \cdot 10^{-88}$	0.00
Waddell – Kappa > PlotCorr	$9.16 \cdot 10^{-54}$	$6.97 \cdot 10^{-61}$
PlotCorr > Waddell – MI	$3.22 \cdot 10^{-83}$	0.00
Waddell – Kappa > MI	$1.79 \cdot 10^{-86}$	$9.83 \cdot 10^{-120}$
MI > Waddell – MI	$6.11 \cdot 10^{-11}$	0.00
Waddell – Kappa > Waddell – MI	$9.37 \cdot 10^{-89}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.45 \cdot 10^{-15}$	$5.31 \cdot 10^{-14}$
CAPS and MI	no difference	no difference
CAPS > Waddell – Kappa	$9.08 \cdot 10^{-07}$	0.00
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$2.04 \cdot 10^{-29}$	0.00
PlotCorr > Waddell – Kappa	$2.41 \cdot 10^{-40}$	0.00
PlotCorr > Waddell – MI	$1.17 \cdot 10^{-06}$	0.01
MI > Waddell – Kappa	no difference	$5.54 \cdot 10^{-05}$
Waddell – MI > MI	$7.98 \cdot 10^{-13}$	$3.79 \cdot 10^{-09}$
Waddell – MI > Waddell – Kappa	$5.05 \cdot 10^{-38}$	$1.19 \cdot 10^{-66}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.13 \cdot 10^{-64}$	$2.07 \cdot 10^{-84}$
MI > CAPS	$2.03 \cdot 10^{-29}$	$5.41 \cdot 10^{-49}$
Waddell – Kappa > CAPS	$1.84 \cdot 10^{-80}$	$3.61 \cdot 10^{-88}$
Waddell – MI > CAPS	$1.87 \cdot 10^{-07}$	0.00
PlotCorr > MI	$1.78 \cdot 10^{-89}$	0.00
Waddell – Kappa > PlotCorr	$2.51 \cdot 10^{-88}$	$1.80 \cdot 10^{-137}$
PlotCorr > Waddell – MI	$1.07 \cdot 10^{-83}$	0.00
Waddell – Kappa > MI	$8.40 \cdot 10^{-90}$	$4.45 \cdot 10^{-162}$
MI > Waddell – MI	$2.79 \cdot 10^{-48}$	0.00
Waddell – Kappa > Waddell – MI	$8.40 \cdot 10^{-90}$	0.00

15–20%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.76 \cdot 10^{-72}$	$2.41 \cdot 10^{-123}$
MI > CAPS	$1.68 \cdot 10^{-06}$	$2.23 \cdot 10^{-06}$
Waddell – Kappa > CAPS	$4.40 \cdot 10^{-72}$	$2.21 \cdot 10^{-119}$
Waddell – MI > CAPS	$8.30 \cdot 10^{-21}$	$5.85 \cdot 10^{-29}$
PlotCorr > MI	$2.52 \cdot 10^{-72}$	0.00
Waddell – Kappa > PlotCorr	$4.72 \cdot 10^{-57}$	$8.70 \cdot 10^{-73}$
PlotCorr > Waddell – MI	$2.76 \cdot 10^{-69}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa > MI	$9.30 \cdot 10^{-72}$	$9.57 \cdot 10^{-116}$
Waddell – MI > MI	$1.27 \cdot 10^{-05}$	$2.61 \cdot 10^{-11}$
Waddell – Kappa > Waddell – MI	$7.00 \cdot 10^{-72}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.96 \cdot 10^{-20}$	$3.51 \cdot 10^{-20}$
CAPS > MI	$1.92 \cdot 10^{-14}$	$1.25 \cdot 10^{-08}$
CAPS > Waddell – Kappa	no difference	0.01
Waddell – MI > CAPS	$8.72 \cdot 10^{-07}$	$5.30 \cdot 10^{-07}$
PlotCorr > MI	$8.61 \cdot 10^{-41}$	0.00
PlotCorr > Waddell – Kappa	$4.86 \cdot 10^{-24}$	0.00
PlotCorr and Waddell – MI	no difference	no difference
Waddell – Kappa > MI	$1.13 \cdot 10^{-10}$	$3.60 \cdot 10^{-05}$
Waddell – MI > MI	$3.29 \cdot 10^{-25}$	$4.15 \cdot 10^{-23}$
Waddell – MI > Waddell – Kappa	$5.68 \cdot 10^{-30}$	$6.68 \cdot 10^{-52}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.20 \cdot 10^{-64}$	$8.22 \cdot 10^{-78}$
MI > CAPS	$7.62 \cdot 10^{-29}$	$2.15 \cdot 10^{-48}$
Waddell – Kappa > CAPS	$1.99 \cdot 10^{-70}$	$5.58 \cdot 10^{-81}$
Waddell – MI > CAPS	$7.78 \cdot 10^{-16}$	$8.34 \cdot 10^{-15}$
PlotCorr > MI	$1.72 \cdot 10^{-72}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$2.50 \cdot 10^{-72}$	$4.78 \cdot 10^{-123}$
PlotCorr > Waddell – MI	$1.01 \cdot 10^{-70}$	0.00
Waddell – Kappa > MI	$1.18 \cdot 10^{-72}$	$1.80 \cdot 10^{-130}$
MI > Waddell – MI	$5.84 \cdot 10^{-41}$	0.00
Waddell – Kappa > Waddell – MI	$2.50 \cdot 10^{-72}$	0.00

20–25%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$4.22 \cdot 10^{-36}$	$1.22 \cdot 10^{-58}$
MI > CAPS	$3.00 \cdot 10^{-09}$	$6.37 \cdot 10^{-09}$
Waddell – Kappa > CAPS	$5.58 \cdot 10^{-37}$	$1.73 \cdot 10^{-60}$
Waddell – MI > CAPS	$1.29 \cdot 10^{-22}$	$2.47 \cdot 10^{-33}$
PlotCorr > MI	$7.92 \cdot 10^{-37}$	0.00
Waddell – Kappa > PlotCorr	$3.01 \cdot 10^{-35}$	$1.20 \cdot 10^{-46}$
PlotCorr > Waddell – MI	$1.31 \cdot 10^{-35}$	0.00
Waddell – Kappa > MI	$8.50 \cdot 10^{-37}$	$2.41 \cdot 10^{-58}$
Waddell – MI > MI	$1.77 \cdot 10^{-05}$	$1.04 \cdot 10^{-08}$
Waddell – Kappa > Waddell – MI	$3.99 \cdot 10^{-37}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.05 \cdot 10^{-14}$	$2.77 \cdot 10^{-12}$
CAPS and MI	no difference	no difference
CAPS and Waddell – Kappa	no difference	no difference
Waddell – MI > CAPS	$1.02 \cdot 10^{-15}$	$5.23 \cdot 10^{-14}$
PlotCorr > MI	$1.54 \cdot 10^{-20}$	$5.88 \cdot 10^{-15}$
PlotCorr > Waddell – Kappa	$1.11 \cdot 10^{-16}$	$1.35 \cdot 10^{-13}$
Waddell – MI > PlotCorr	$2.28 \cdot 10^{-05}$	$4.48 \cdot 10^{-06}$
Waddell – Kappa > MI	$3.17 \cdot 10^{-06}$	0.02
Waddell – MI > MI	$1.23 \cdot 10^{-21}$	$9.79 \cdot 10^{-20}$
Waddell – MI > Waddell – Kappa	$9.96 \cdot 10^{-24}$	$1.84 \cdot 10^{-40}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$5.67 \cdot 10^{-29}$	$8.80 \cdot 10^{-25}$
MI > CAPS	$2.04 \cdot 10^{-05}$	$1.13 \cdot 10^{-13}$
Waddell – Kappa > CAPS	$2.38 \cdot 10^{-33}$	$1.70 \cdot 10^{-25}$
Waddell – MI > CAPS	no difference	0.02
PlotCorr > MI	$3.67 \cdot 10^{-37}$	0.00
Waddell – Kappa > PlotCorr	$3.88 \cdot 10^{-37}$	$8.74 \cdot 10^{-61}$
PlotCorr > Waddell – MI	$1.49 \cdot 10^{-36}$	0.00
Waddell – Kappa > MI	$3.67 \cdot 10^{-37}$	$3.80 \cdot 10^{-65}$
MI > Waddell – MI	$2.08 \cdot 10^{-29}$	0.00
Waddell – Kappa > Waddell – MI	$3.67 \cdot 10^{-37}$	0.00

25–30%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.72 \cdot 10^{-75}$	$1.55 \cdot 10^{-133}$
MI > CAPS	$1.93 \cdot 10^{-10}$	$5.22 \cdot 10^{-17}$
Waddell – Kappa > CAPS	$1.11 \cdot 10^{-74}$	$5.90 \cdot 10^{-122}$
Waddell – MI > CAPS	$3.49 \cdot 10^{-20}$	$1.78 \cdot 10^{-24}$
PlotCorr > MI	$4.10 \cdot 10^{-75}$	0.00
Waddell – Kappa > PlotCorr	$9.07 \cdot 10^{-33}$	$5.89 \cdot 10^{-29}$
PlotCorr > Waddell – MI	$2.77 \cdot 10^{-75}$	0.00
Waddell – Kappa > MI	$1.20 \cdot 10^{-71}$	$6.72 \cdot 10^{-105}$
MI and Waddell – MI	no difference	no difference
Waddell – Kappa > Waddell – MI	$6.63 \cdot 10^{-72}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$3.28 \cdot 10^{-44}$	$6.50 \cdot 10^{-54}$
MI > CAPS	no difference	0.00
CAPS > Waddell – Kappa	$7.11 \cdot 10^{-07}$	$1.34 \cdot 10^{-05}$
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$5.60 \cdot 10^{-21}$	$2.26 \cdot 10^{-10}$
PlotCorr > Waddell – Kappa	$1.58 \cdot 10^{-66}$	$-2.22 \cdot 10^{-16}$
PlotCorr > Waddell – MI	$2.41 \cdot 10^{-19}$	$4.54 \cdot 10^{-08}$
MI > Waddell – Kappa	$1.32 \cdot 10^{-09}$	$1.28 \cdot 10^{-05}$
Waddell – MI > MI	0.00	0.00
Waddell – MI > Waddell – Kappa	$1.09 \cdot 10^{-36}$	$3.04 \cdot 10^{-64}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.12 \cdot 10^{-57}$	$1.63 \cdot 10^{-30}$
MI > CAPS	no difference	$8.03 \cdot 10^{-06}$
Waddell – Kappa > CAPS	$5.71 \cdot 10^{-62}$	$1.61 \cdot 10^{-42}$
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$9.28 \cdot 10^{-76}$	0.00
Waddell – Kappa > PlotCorr	$2.49 \cdot 10^{-73}$	$6.38 \cdot 10^{-114}$
PlotCorr > Waddell – MI	$3.00 \cdot 10^{-73}$	0.00
Waddell – Kappa > MI	$9.28 \cdot 10^{-76}$	$3.44 \cdot 10^{-136}$
MI > Waddell – MI	$7.47 \cdot 10^{-35}$	0.00
Waddell – Kappa > Waddell – MI	$4.18 \cdot 10^{-75}$	0.00

30–35%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$6.19 \cdot 10^{-18}$	$6.31 \cdot 10^{-30}$
MI > CAPS	$1.02 \cdot 10^{-12}$	$1.04 \cdot 10^{-13}$
Waddell – Kappa > CAPS	$6.18 \cdot 10^{-18}$	$6.31 \cdot 10^{-30}$
Waddell – MI > CAPS	$4.24 \cdot 10^{-13}$	$9.16 \cdot 10^{-15}$
PlotCorr > MI	$7.94 \cdot 10^{-18}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$2.34 \cdot 10^{-17}$	$3.00 \cdot 10^{-26}$
PlotCorr > Waddell – MI	$1.08 \cdot 10^{-17}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$6.19 \cdot 10^{-18}$	$6.31 \cdot 10^{-30}$
Waddell – MI > MI	0.01	0.00
Waddell – Kappa > Waddell – MI	$6.38 \cdot 10^{-18}$	$1.11 \cdot 10^{-16}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$9.01 \cdot 10^{-13}$	$2.74 \cdot 10^{-11}$
MI > CAPS	$1.80 \cdot 10^{-05}$	0.00
CAPS and Waddell – Kappa	no difference	no difference
Waddell – MI > CAPS	$1.13 \cdot 10^{-09}$	0.00
PlotCorr > MI	$4.88 \cdot 10^{-12}$	$2.83 \cdot 10^{-07}$
PlotCorr > Waddell – Kappa	$2.19 \cdot 10^{-08}$	$4.31 \cdot 10^{-09}$
PlotCorr and Waddell – MI	no difference	no difference
MI and Waddell – Kappa	no difference	no difference
Waddell – MI > MI	$1.87 \cdot 10^{-10}$	$2.47 \cdot 10^{-06}$
Waddell – MI > Waddell – Kappa	$6.72 \cdot 10^{-14}$	$2.12 \cdot 10^{-22}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.47 \cdot 10^{-08}$	0.00
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	$1.26 \cdot 10^{-10}$	0.00
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$6.58 \cdot 10^{-18}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$6.37 \cdot 10^{-18}$	$6.18 \cdot 10^{-28}$
PlotCorr > Waddell – MI	$1.72 \cdot 10^{-17}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > MI	$6.18 \cdot 10^{-18}$	$6.31 \cdot 10^{-30}$
MI > Waddell – MI	$8.78 \cdot 10^{-13}$	$5.77 \cdot 10^{-10}$
Waddell – Kappa > Waddell – MI	$6.18 \cdot 10^{-18}$	$1.11 \cdot 10^{-16}$

>35%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
MI > CAPS	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
Waddell – Kappa > CAPS	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
Waddell – MI > CAPS	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
PlotCorr > MI	$1.78 \cdot 10^{-15}$	$1.67 \cdot 10^{-15}$
Waddell – Kappa > PlotCorr	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
PlotCorr > Waddell – MI	$6.88 \cdot 10^{-09}$	$2.86 \cdot 10^{-08}$
Waddell – Kappa > MI	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
Waddell – MI > MI	$8.06 \cdot 10^{-11}$	$2.18 \cdot 10^{-12}$
Waddell – Kappa > Waddell – MI	$1.78 \cdot 10^{-15}$	$1.67 \cdot 10^{-15}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
CAPS and PlotCorr	no difference	no difference
CAPS and MI	no difference	no difference
CAPS > Waddell – Kappa	no difference	0.00
CAPS and Waddell – MI	no difference	no difference
PlotCorr and MI	no difference	no difference
PlotCorr > Waddell – Kappa	$3.07 \cdot 10^{-06}$	$1.92 \cdot 10^{-05}$
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	no difference	0.02
MI and Waddell – MI	no difference	no difference
Waddell – MI > Waddell – Kappa	$3.94 \cdot 10^{-09}$	$5.68 \cdot 10^{-14}$

Specificities

Methods	Wilcoxon p-value	Sign p-value
PlotCorr > CAPS	$1.36 \cdot 10^{-08}$	$4.63 \cdot 10^{-06}$
MI > CAPS	0.01	$4.63 \cdot 10^{-06}$
Waddell – Kappa > CAPS	$2.99 \cdot 10^{-11}$	$4.63 \cdot 10^{-06}$
Waddell – MI > CAPS	0.01	$4.63 \cdot 10^{-06}$
PlotCorr > MI	$1.78 \cdot 10^{-15}$	$1.67 \cdot 10^{-15}$
Waddell – Kappa > PlotCorr	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
PlotCorr > Waddell – MI	$2.52 \cdot 10^{-10}$	$2.86 \cdot 10^{-08}$
Waddell – Kappa > MI	$1.78 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$
Waddell – MI > MI	0.01	0.00
Waddell – Kappa > Waddell – MI	$1.78 \cdot 10^{-15}$	$1.67 \cdot 10^{-15}$

"Higher" UAA threshold

0–5%

Precisions

Methods	Wilcoxon p-value	Sign p-value
PlotCorr > CAPS	$1.76 \cdot 10^{-71}$	$7.39 \cdot 10^{-116}$
MI > CAPS	$6.13 \cdot 10^{-79}$	$2.72 \cdot 10^{-137}$
Waddell – Kappa > CAPS	$3.08 \cdot 10^{-76}$	$2.94 \cdot 10^{-123}$
Waddell – MI > CAPS	$6.63 \cdot 10^{-82}$	$2.59 \cdot 10^{-134}$
PlotCorr > MI	$8.84 \cdot 10^{-62}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$5.63 \cdot 10^{-72}$	$2.13 \cdot 10^{-113}$
Waddell – MI > PlotCorr	$4.59 \cdot 10^{-80}$	$1.66 \cdot 10^{-127}$
Waddell – Kappa > MI	$7.03 \cdot 10^{-69}$	$4.29 \cdot 10^{-98}$
Waddell – MI > MI	$1.89 \cdot 10^{-80}$	$1.57 \cdot 10^{-125}$
Waddell – Kappa > Waddell – MI	$1.01 \cdot 10^{-26}$	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$8.66 \cdot 10^{-31}$	$2.26 \cdot 10^{-25}$
MI > CAPS	$3.68 \cdot 10^{-56}$	$3.03 \cdot 10^{-75}$
Waddell – Kappa > CAPS	$1.27 \cdot 10^{-34}$	$4.79 \cdot 10^{-28}$
Waddell – MI > CAPS	$7.45 \cdot 10^{-42}$	$2.51 \cdot 10^{-34}$
MI > PlotCorr	$1.09 \cdot 10^{-25}$	$6.23 \cdot 10^{-43}$
PlotCorr and Waddell – Kappa	no difference	no difference
Waddell – MI > PlotCorr	$1.46 \cdot 10^{-05}$	$5.81 \cdot 10^{-05}$
MI > Waddell – Kappa	$8.16 \cdot 10^{-19}$	0.00
MI > Waddell – MI	$4.12 \cdot 10^{-12}$	$-2.22 \cdot 10^{-16}$
Waddell – MI > Waddell – Kappa	$1.72 \cdot 10^{-11}$	$5.95 \cdot 10^{-14}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.10 \cdot 10^{-32}$	0.00
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	$1.19 \cdot 10^{-39}$	0.00
Waddell – MI > CAPS	$2.04 \cdot 10^{-39}$	0.00
PlotCorr > MI	$3.44 \cdot 10^{-95}$	0.00
Waddell – Kappa > PlotCorr	$3.44 \cdot 10^{-95}$	$5.18 \cdot 10^{-172}$
Waddell – MI > PlotCorr	$5.45 \cdot 10^{-95}$	$3.16 \cdot 10^{-164}$
Waddell – Kappa > MI	$3.44 \cdot 10^{-95}$	$5.18 \cdot 10^{-172}$
Waddell – MI > MI	$3.44 \cdot 10^{-95}$	$5.18 \cdot 10^{-172}$
Waddell – Kappa > Waddell – MI	$1.02 \cdot 10^{-62}$	0.00

5–10%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.32 \cdot 10^{-17}$	$2.52 \cdot 10^{-29}$
MI > CAPS	$2.74 \cdot 10^{-17}$	$8.40 \cdot 10^{-23}$
Waddell – Kappa > CAPS	$1.93 \cdot 10^{-17}$	$5.05 \cdot 10^{-29}$
Waddell – MI > CAPS	$1.93 \cdot 10^{-17}$	$5.05 \cdot 10^{-29}$
PlotCorr > MI	$2.06 \cdot 10^{-15}$	0.00
Waddell – Kappa > PlotCorr	$3.53 \cdot 10^{-17}$	$1.15 \cdot 10^{-25}$
Waddell – MI > PlotCorr	$1.57 \cdot 10^{-16}$	$3.61 \cdot 10^{-24}$
Waddell – Kappa > MI	$2.62 \cdot 10^{-17}$	$1.86 \cdot 10^{-24}$
Waddell – MI > MI	$2.24 \cdot 10^{-17}$	$1.86 \cdot 10^{-24}$
Waddell – Kappa > Waddell – MI	0.00	0.00

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$1.16 \cdot 10^{-10}$	$7.07 \cdot 10^{-09}$
MI > CAPS	$7.48 \cdot 10^{-14}$	$3.05 \cdot 10^{-20}$
Waddell – Kappa > CAPS	$1.12 \cdot 10^{-09}$	$5.21 \cdot 10^{-07}$
Waddell – MI > CAPS	$3.75 \cdot 10^{-11}$	$1.15 \cdot 10^{-08}$
MI > PlotCorr	$8.71 \cdot 10^{-09}$	$5.89 \cdot 10^{-13}$
PlotCorr and Waddell – Kappa	no difference	no difference
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$9.20 \cdot 10^{-08}$	$1.85 \cdot 10^{-11}$
MI > Waddell – MI	$5.02 \cdot 10^{-06}$	$5.24 \cdot 10^{-09}$
Waddell – MI > Waddell – Kappa	0.00	$3.05 \cdot 10^{-05}$

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.89 \cdot 10^{-09}$	0.01
CAPS and MI	no difference	no difference
Waddell – Kappa > CAPS	$6.33 \cdot 10^{-10}$	0.01
Waddell – MI > CAPS	$6.33 \cdot 10^{-10}$	0.01
PlotCorr > MI	$9.33 \cdot 10^{-18}$	0.00
Waddell – Kappa > PlotCorr	$9.03 \cdot 10^{-18}$	$1.26 \cdot 10^{-29}$
Waddell – MI > PlotCorr	$9.63 \cdot 10^{-18}$	$1.22 \cdot 10^{-27}$
Waddell – Kappa > MI	$9.04 \cdot 10^{-18}$	$1.26 \cdot 10^{-29}$
Waddell – MI > MI	$9.04 \cdot 10^{-18}$	$1.26 \cdot 10^{-29}$
Waddell – Kappa > Waddell – MI	$6.51 \cdot 10^{-06}$	$7.31 \cdot 10^{-08}$

10–15%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.73 \cdot 10^{-08}$	$1.82 \cdot 10^{-12}$
MI > CAPS	$4.66 \cdot 10^{-07}$	$7.28 \cdot 10^{-11}$
Waddell – Kappa > CAPS	$4.35 \cdot 10^{-08}$	$1.42 \cdot 10^{-09}$
Waddell – MI > CAPS	$4.02 \cdot 10^{-08}$	$7.28 \cdot 10^{-11}$
PlotCorr > MI	$1.82 \cdot 10^{-12}$	$1.82 \cdot 10^{-12}$
Waddell – Kappa > PlotCorr	$1.84 \cdot 10^{-06}$	$1.80 \cdot 10^{-08}$
Waddell – MI > PlotCorr	$7.81 \cdot 10^{-07}$	$1.21 \cdot 10^{-06}$
Waddell – Kappa > MI	$3.77 \cdot 10^{-10}$	$7.28 \cdot 10^{-11}$
Waddell – MI > MI	$1.82 \cdot 10^{-12}$	$1.82 \cdot 10^{-12}$
Waddell – Kappa and Waddell – MI	no difference	no difference

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$6.36 \cdot 10^{-06}$	0.00
MI > CAPS	$6.74 \cdot 10^{-07}$	$1.86 \cdot 10^{-09}$
Waddell – Kappa > CAPS	0.00	0.01
Waddell – MI > CAPS	$6.97 \cdot 10^{-05}$	0.00
MI > PlotCorr	0.00	$3.30 \cdot 10^{-05}$
PlotCorr and Waddell – Kappa	no difference	no difference
PlotCorr and Waddell – MI	no difference	no difference
MI > Waddell – Kappa	$2.69 \cdot 10^{-05}$	$7.75 \cdot 10^{-07}$
MI > Waddell – MI	0.00	0.00
Waddell – MI > Waddell – Kappa	0.04	0.02

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
CAPS and PlotCorr	no difference	no difference
CAPS and MI	no difference	no difference
CAPS and Waddell – Kappa	no difference	no difference
CAPS and Waddell – MI	no difference	no difference
PlotCorr > MI	$2.73 \cdot 10^{-08}$	$1.82 \cdot 10^{-12}$
Waddell – Kappa > PlotCorr	$2.73 \cdot 10^{-08}$	$1.82 \cdot 10^{-12}$
Waddell – MI > PlotCorr	$2.95 \cdot 10^{-08}$	$7.28 \cdot 10^{-11}$
Waddell – Kappa > MI	$2.73 \cdot 10^{-08}$	$1.82 \cdot 10^{-12}$
Waddell – MI > MI	$2.73 \cdot 10^{-08}$	$1.82 \cdot 10^{-12}$
Waddell – Kappa and Waddell – MI	no difference	no difference

>15%

Precisions

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
MI > CAPS	$2.74 \cdot 10^{-10}$	$7.71 \cdot 10^{-13}$
Waddell – Kappa > CAPS	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
Waddell – MI > CAPS	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
PlotCorr > MI	$5.69 \cdot 10^{-11}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa > PlotCorr	$8.41 \cdot 10^{-07}$	$4.03 \cdot 10^{-08}$
Waddell – MI > PlotCorr	0.01	0.00
Waddell – Kappa > MI	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
Waddell – MI > MI	$1.52 \cdot 10^{-10}$	$4.28 \cdot 10^{-14}$
Waddell – Kappa > Waddell – MI	$3.34 \cdot 10^{-06}$	$9.90 \cdot 10^{-05}$

Sensitivities

Methods	**Wilcoxon p-value**	**Sign p-value**
PlotCorr > CAPS	$2.99 \cdot 10^{-09}$	$6.10 \cdot 10^{-07}$
MI > CAPS	$8.96 \cdot 10^{-11}$	$1.11 \cdot 10^{-16}$
Waddell – Kappa > CAPS	0.01	0.00
Waddell – MI > CAPS	0.01	0.00
MI > PlotCorr	$4.21 \cdot 10^{-09}$	$2.56 \cdot 10^{-12}$
PlotCorr > Waddell – Kappa	$3.11 \cdot 10^{-08}$	$3.53 \cdot 10^{-11}$
PlotCorr > Waddell – MI	$7.17 \cdot 10^{-08}$	$1.18 \cdot 10^{-07}$
MI > Waddell – Kappa	$1.78 \cdot 10^{-10}$	$5.55 \cdot 10^{-16}$
MI > Waddell – MI	$1.79 \cdot 10^{-10}$	$5.55 \cdot 10^{-16}$
Waddell – Kappa and Waddell – MI	no difference	no difference

Specificities

Methods	**Wilcoxon p-value**	**Sign p-value**
CAPS and PlotCorr	no difference	no difference
CAPS $>$ MI	$9.47 \cdot 10^{-05}$	0.00
CAPS and Waddell – Kappa	no difference	no difference
CAPS and Waddell – MI	no difference	no difference
PlotCorr $>$ MI	$5.69 \cdot 10^{-11}$	$-2.22 \cdot 10^{-16}$
Waddell – Kappa $>$ PlotCorr	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
Waddell – MI $>$ PlotCorr	$2.22 \cdot 10^{-10}$	$7.71 \cdot 10^{-13}$
Waddell – Kappa $>$ MI	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
Waddell – MI $>$ MI	$5.69 \cdot 10^{-11}$	$2.78 \cdot 10^{-17}$
Waddell – Kappa $>$ Waddell – MI	$1.13 \cdot 10^{-05}$	$2.85 \cdot 10^{-05}$

A.9.2 Comparing percentages of sites co-evolving

In this section, we compare the performance of each method for different percentages of sites co-evolving, for the “lower” and “higher” unique amino acid thresholds.

“Lower” UAA threshold

CAPS

Precisions

Methods	Wilcoxon p-value
10 $>$ 5	$1.06 \cdot 10^{-110}$
15 $>$ 5	$3.93 \cdot 10^{-124}$
20 $>$ 5	$9.19 \cdot 10^{-101}$
25 $>$ 5	$9.58 \cdot 10^{-44}$
30 $>$ 5	$1.11 \cdot 10^{-84}$
35 $>$ 5	$4.06 \cdot 10^{-08}$
greater than 35 $>$ 5	$9.04 \cdot 10^{-15}$
15 $>$ 10	$1.76 \cdot 10^{-13}$
20 $>$ 10	$1.77 \cdot 10^{-05}$
25 $>$ 10	0.00
30 $>$ 10	$3.70 \cdot 10^{-36}$
10 and 35	no difference
greater than 35 $>$ 10	$1.01 \cdot 10^{-05}$
15 $>$ 20	0.01
15 and 25	no difference
30 $>$ 15	$1.41 \cdot 10^{-15}$
15 $>$ 35	0.00
15 and greater than 35	no difference
20 and 25	no difference
20 and 35	no difference
greater than 35 $>$ 20	0.02
30 $>$ 25	$2.61 \cdot 10^{-10}$
25 and 35	no difference
25 and greater than 35	no difference
30 $>$ 35	$2.73 \cdot 10^{-10}$
30 $>$ greater than 35	0.00
greater than 35 $>$ 35	0.01

Sensitivities

Methods	**Wilcoxon p-value**
5 and 10	no difference
5 and 15	no difference
5 $>$ 20	$3.87 \cdot 10^{-12}$
5 $>$ 25	$9.58 \cdot 10^{-19}$
5 $>$ 30	$1.53 \cdot 10^{-19}$
5 $>$ 35	$1.89 \cdot 10^{-17}$
5 $>$ greater than 35	$1.70 \cdot 10^{-08}$
10 and 15	no difference
10 $>$ 20	$2.57 \cdot 10^{-13}$
10 $>$ 25	$2.88 \cdot 10^{-21}$
10 $>$ 30	$4.40 \cdot 10^{-20}$
10 $>$ 35	$9.05 \cdot 10^{-19}$
10 $>$ greater than 35	$1.95 \cdot 10^{-09}$
15 $>$ 20	$2.70 \cdot 10^{-11}$
15 $>$ 25	$3.98 \cdot 10^{-19}$
15 $>$ 30	$6.38 \cdot 10^{-18}$
15 $>$ 35	$4.05 \cdot 10^{-17}$
15 $>$ greater than 35	$1.11 \cdot 10^{-07}$
20 $>$ 25	$8.77 \cdot 10^{-05}$
20 $>$ 30	0.00
20 $>$ 35	$5.13 \cdot 10^{-06}$
20 and greater than 35	no difference
25 and 30	no difference
25 and 35	no difference
25 and greater than 35	no difference
30 $>$ 35	0.01
30 and greater than 35	no difference
35 and greater than 35	no difference

Specificities

Methods	Wilcoxon p-value
$5 > 10$	0.00
$5 > 15$	$8.12 \cdot 10^{-15}$
$5 > 20$	$1.26 \cdot 10^{-54}$
$5 > 25$	$1.40 \cdot 10^{-15}$
$5 > 30$	$9.09 \cdot 10^{-18}$
5 and 35	no difference
$5 >$ greater than 35	0.00
$10 > 15$	$5.08 \cdot 10^{-06}$
$10 > 20$	$6.19 \cdot 10^{-30}$
$10 > 25$	$9.59 \cdot 10^{-09}$
$10 > 30$	$6.00 \cdot 10^{-07}$
10 and 35	no difference
$10 >$ greater than 35	0.00
$15 > 20$	$3.44 \cdot 10^{-13}$
$15 > 25$	0.00
15 and 30	no difference
$35 > 15$	0.00
15 and greater than 35	no difference
$25 > 20$	0.01
$30 > 20$	$1.78 \cdot 10^{-14}$
$35 > 20$	$1.29 \cdot 10^{-11}$
greater than $35 > 20$	0.00
$30 > 25$	0.00
$35 > 25$	$7.13 \cdot 10^{-06}$
greater than $35 > 25$	0.02
$35 > 30$	0.00
30 and greater than 35	no difference
35 and greater than 35	no difference

PlotCorr

Precisions

Methods	**Wilcoxon p-value**
10 > 5	$1.17 \cdot 10^{-213}$
15 > 5	$2.85 \cdot 10^{-205}$
20 > 5	$9.26 \cdot 10^{-200}$
25 > 5	$1.50 \cdot 10^{-111}$
30 > 5	$8.12 \cdot 10^{-248}$
35 > 5	$9.80 \cdot 10^{-60}$
greater than 35 > 5	$5.73 \cdot 10^{-32}$
15 > 10	$4.57 \cdot 10^{-28}$
20 > 10	$1.18 \cdot 10^{-35}$
25 > 10	$6.53 \cdot 10^{-45}$
30 > 10	$3.75 \cdot 10^{-158}$
35 > 10	$2.52 \cdot 10^{-28}$
greater than 35 > 10	$2.53 \cdot 10^{-15}$
15 and 20	no difference
25 > 15	$1.80 \cdot 10^{-09}$
30 > 15	$1.55 \cdot 10^{-96}$
35 > 15	$5.11 \cdot 10^{-05}$
greater than 35 > 15	0.00
25 > 20	$2.99 \cdot 10^{-09}$
30 > 20	$9.72 \cdot 10^{-96}$
35 > 20	$1.70 \cdot 10^{-05}$
greater than 35 > 20	0.00
30 > 25	$7.41 \cdot 10^{-45}$
25 and 35	no difference
25 and greater than 35	no difference
30 > 35	$4.48 \cdot 10^{-23}$
30 > greater than 35	$3.25 \cdot 10^{-15}$
35 and greater than 35	no difference

Sensitivities

Methods	**Wilcoxon p-value**
$5 > 10$	0.00
$5 > 15$	0.00
$5 > 20$	$4.20 \cdot 10^{-25}$
$5 > 25$	$1.58 \cdot 10^{-29}$
$5 > 30$	$6.70 \cdot 10^{-20}$
$5 > 35$	$3.86 \cdot 10^{-14}$
$5 >$ greater than 35	$2.36 \cdot 10^{-20}$
10 and 15	no difference
$10 > 20$	$2.53 \cdot 10^{-13}$
$10 > 25$	$3.92 \cdot 10^{-20}$
$10 > 30$	$7.77 \cdot 10^{-10}$
$10 > 35$	$3.72 \cdot 10^{-09}$
$10 >$ greater than 35	$1.38 \cdot 10^{-14}$
$15 > 20$	$5.19 \cdot 10^{-11}$
$15 > 25$	$1.29 \cdot 10^{-17}$
$15 > 30$	$6.21 \cdot 10^{-08}$
$15 > 35$	$3.09 \cdot 10^{-08}$
$15 >$ greater than 35	$5.61 \cdot 10^{-13}$
$20 > 25$	0.00
20 and 30	no difference
20 and 35	no difference
$20 >$ greater than 35	0.01
$30 > 25$	$1.57 \cdot 10^{-06}$
$35 > 25$	0.00
25 and greater than 35	no difference
30 and 35	no difference
$30 >$ greater than 35	0.00
$35 >$ greater than 35	0.01

Specificities

Methods	**Wilcoxon p-value**
5 and 10	no difference
$5 > 15$	0.01
$5 > 20$	$1.66 \cdot 10^{-13}$
5 and 25	no difference
$30 > 5$	$7.50 \cdot 10^{-62}$
$35 > 5$	0.02
5 and greater than 35	no difference
10 and 15	no difference
$10 > 20$	$2.77 \cdot 10^{-07}$
10 and 25	no difference
$30 > 10$	$6.67 \cdot 10^{-44}$
10 and 35	no difference
10 and greater than 35	no difference
$15 > 20$	0.00
15 and 25	no difference
$30 > 15$	$4.56 \cdot 10^{-34}$
$35 > 15$	0.01
15 and greater than 35	no difference
$25 > 20$	$6.44 \cdot 10^{-07}$
$30 > 20$	$1.52 \cdot 10^{-59}$
$35 > 20$	$1.58 \cdot 10^{-07}$
greater than $35 > 20$	0.01
$30 > 25$	$7.03 \cdot 10^{-22}$
25 and 35	no difference
25 and greater than 35	no difference
$30 > 35$	$1.96 \cdot 10^{-11}$
$30 >$ greater than 35	$4.07 \cdot 10^{-11}$
35 and greater than 35	no difference

MI

Precisions

Methods	**Wilcoxon p-value**
$10 > 5$	$2.91 \cdot 10^{-127}$
$15 > 5$	$3.02 \cdot 10^{-99}$
$20 > 5$	$3.96 \cdot 10^{-68}$
$25 > 5$	$5.48 \cdot 10^{-64}$
$30 > 5$	$3.46 \cdot 10^{-71}$
$35 > 5$	$2.56 \cdot 10^{-46}$
greater than $35 > 5$	$1.35 \cdot 10^{-28}$
$15 > 10$	$3.81 \cdot 10^{-07}$
10 and 20	no difference
$25 > 10$	$6.22 \cdot 10^{-05}$
$30 > 10$	$2.49 \cdot 10^{-22}$
$35 > 10$	$5.01 \cdot 10^{-06}$
greater than $35 > 10$	$4.65 \cdot 10^{-11}$
$15 > 20$	0.00
15 and 25	no difference
$30 > 15$	$6.79 \cdot 10^{-11}$
15 and 35	no difference
greater than $35 > 15$	0.00
$25 > 20$	0.00
$30 > 20$	$1.20 \cdot 10^{-18}$
$35 > 20$	$2.16 \cdot 10^{-07}$
greater than $35 > 20$	$1.20 \cdot 10^{-08}$
$30 > 25$	$8.07 \cdot 10^{-06}$
$35 > 25$	0.02
greater than $35 > 25$	0.00
$30 > 35$	0.02
30 and greater than 35	no difference
greater than $35 > 35$	0.01

Sensitivities

Methods	**Wilcoxon p-value**
5 and 10	no difference
5 and 15	no difference
5 $>$ 20	$2.15 \cdot 10^{-26}$
5 $>$ 25	$1.82 \cdot 10^{-20}$
5 $>$ 30	$3.81 \cdot 10^{-15}$
5 $>$ 35	$1.74 \cdot 10^{-06}$
5 $>$ greater than 35	$3.85 \cdot 10^{-05}$
10 and 15	no difference
10 $>$ 20	$6.11 \cdot 10^{-23}$
10 $>$ 25	$4.80 \cdot 10^{-20}$
10 $>$ 30	$6.88 \cdot 10^{-12}$
10 $>$ 35	$9.13 \cdot 10^{-06}$
10 $>$ greater than 35	0.00
15 $>$ 20	$4.71 \cdot 10^{-14}$
15 $>$ 25	$3.75 \cdot 10^{-13}$
15 $>$ 30	$1.33 \cdot 10^{-06}$
15 $>$ 35	0.00
15 $>$ greater than 35	0.01
20 and 25	no difference
30 $>$ 20	0.01
35 $>$ 20	0.00
greater than 35 $>$ 20	0.00
30 $>$ 25	0.00
35 $>$ 25	$4.55 \cdot 10^{-05}$
greater than 35 $>$ 25	$4.88 \cdot 10^{-06}$
30 and 35	no difference
30 and greater than 35	no difference
greater than 35 $>$ 35	0.02

Specificities

Methods	Wilcoxon p-value
$10 > 5$	0.00
5 and 15	no difference
$5 > 20$	$2.04 \cdot 10^{-22}$
$5 > 25$	$2.81 \cdot 10^{-13}$
$5 > 30$	$4.24 \cdot 10^{-22}$
$5 > 35$	0.00
5 and greater than 35	no difference
$10 > 15$	0.02
$10 > 20$	$6.32 \cdot 10^{-26}$
$10 > 25$	$7.18 \cdot 10^{-17}$
$10 > 30$	$1.74 \cdot 10^{-25}$
$10 > 35$	$1.56 \cdot 10^{-06}$
10 and greater than 35	no difference
$15 > 20$	$5.22 \cdot 10^{-16}$
$15 > 25$	$1.02 \cdot 10^{-12}$
$15 > 30$	$1.26 \cdot 10^{-13}$
$15 > 35$	$6.46 \cdot 10^{-05}$
15 and greater than 35	no difference
20 and 25	no difference
20 and 30	no difference
20 and 35	no difference
greater than $35 > 20$	$5.28 \cdot 10^{-09}$
$30 > 25$	0.00
25 and 35	no difference
greater than $35 > 25$	$4.20 \cdot 10^{-08}$
30 and 35	no difference
greater than $35 > 30$	$1.59 \cdot 10^{-08}$
greater than $35 > 35$	0.00

Waddell – Kappa

Precisions

Methods	**Wilcoxon p-value**
$10 > 5$	$5.14 \cdot 10^{-202}$
$15 > 5$	$4.00 \cdot 10^{-208}$
$20 > 5$	$1.65 \cdot 10^{-209}$
$25 > 5$	$1.37 \cdot 10^{-115}$
$30 > 5$	$9.18 \cdot 10^{-234}$
$35 > 5$	$3.88 \cdot 10^{-61}$
greater than $35 > 5$	$2.15 \cdot 10^{-32}$
$15 > 10$	$6.83 \cdot 10^{-35}$
$20 > 10$	$2.21 \cdot 10^{-61}$
$25 > 10$	$7.08 \cdot 10^{-61}$
$30 > 10$	$6.07 \cdot 10^{-155}$
$35 > 10$	$1.56 \cdot 10^{-39}$
greater than $35 > 10$	$4.09 \cdot 10^{-19}$
$20 > 15$	$1.96 \cdot 10^{-05}$
$25 > 15$	$9.46 \cdot 10^{-20}$
$30 > 15$	$2.64 \cdot 10^{-96}$
$35 > 15$	$4.26 \cdot 10^{-15}$
greater than $35 > 15$	$5.06 \cdot 10^{-05}$
$25 > 20$	$3.82 \cdot 10^{-12}$
$30 > 20$	$8.29 \cdot 10^{-80}$
$35 > 20$	$9.65 \cdot 10^{-10}$
greater than $35 > 20$	0.01
$30 > 25$	$7.59 \cdot 10^{-30}$
25 and 35	no difference
$25 >$ greater than 35	0.00
$30 > 35$	$5.73 \cdot 10^{-17}$
$30 >$ greater than 35	$8.43 \cdot 10^{-13}$
$35 >$ greater than 35	0.00

Sensitivities

Methods	**Wilcoxon p-value**
$5 > 10$	$8.61 \cdot 10^{-06}$
$5 > 15$	$2.07 \cdot 10^{-13}$
$5 > 20$	$6.20 \cdot 10^{-19}$
$5 > 25$	$4.91 \cdot 10^{-14}$
$5 > 30$	$2.52 \cdot 10^{-77}$
$5 > 35$	$3.79 \cdot 10^{-05}$
$5 >$ greater than 35	0.00
$10 > 15$	$7.19 \cdot 10^{-05}$
$10 > 20$	$2.12 \cdot 10^{-10}$
$10 > 25$	$3.38 \cdot 10^{-09}$
$10 > 30$	$1.97 \cdot 10^{-73}$
10 and 35	no difference
10 and greater than 35	no difference
$15 > 20$	0.01
$15 > 25$	0.00
$15 > 30$	$4.45 \cdot 10^{-43}$
$35 > 15$	0.02
greater than $35 > 15$	0.02
20 and 25	no difference
$20 > 30$	$8.86 \cdot 10^{-29}$
$35 > 20$	$4.61 \cdot 10^{-05}$
greater than $35 > 20$	0.00
$25 > 30$	$4.12 \cdot 10^{-12}$
$35 > 25$	$5.72 \cdot 10^{-05}$
greater than $35 > 25$	0.00
$35 > 30$	$5.53 \cdot 10^{-30}$
greater than $35 > 30$	$2.62 \cdot 10^{-19}$
35 and greater than 35	no difference

Specificities

Methods	**Wilcoxon p-value**
5 and 10	no difference
5 and 15	no difference
$20 > 5$	0.00
$25 > 5$	$4.17 \cdot 10^{-21}$
$30 > 5$	$2.13 \cdot 10^{-82}$
$35 > 5$	$2.18 \cdot 10^{-13}$
5 and greater than 35	no difference
10 and 15	no difference
$20 > 10$	0.00
$25 > 10$	$7.44 \cdot 10^{-19}$
$30 > 10$	$8.47 \cdot 10^{-64}$
$35 > 10$	$5.04 \cdot 10^{-13}$
10 and greater than 35	no difference
15 and 20	no difference
$25 > 15$	$5.05 \cdot 10^{-12}$
$30 > 15$	$1.83 \cdot 10^{-45}$
$35 > 15$	$4.37 \cdot 10^{-09}$
15 and greater than 35	no difference
$25 > 20$	$1.67 \cdot 10^{-10}$
$30 > 20$	$1.24 \cdot 10^{-41}$
$35 > 20$	$2.63 \cdot 10^{-09}$
20 and greater than 35	no difference
$30 > 25$	$6.29 \cdot 10^{-11}$
25 and 35	no difference
$25 >$ greater than 35	$2.27 \cdot 10^{-05}$
$30 > 35$	$1.48 \cdot 10^{-05}$
$30 >$ greater than 35	$1.81 \cdot 10^{-09}$
$35 >$ greater than 35	$2.96 \cdot 10^{-05}$

Waddell – MI

Precisions

Methods	Wilcoxon p-value
$10 > 5$	$5.64 \cdot 10^{-210}$
$15 > 5$	$2.69 \cdot 10^{-205}$
$20 > 5$	$1.46 \cdot 10^{-191}$
$25 > 5$	$1\,0.00 \cdot 10^{-111}$
$30 > 5$	$9.16 \cdot 10^{-245}$
$35 > 5$	$6.65 \cdot 10^{-57}$
greater than $35 > 5$	$8.26 \cdot 10^{-32}$
$15 > 10$	$2.10 \cdot 10^{-23}$
$20 > 10$	$2.16 \cdot 10^{-22}$
$25 > 10$	$2.80 \cdot 10^{-27}$
$30 > 10$	$1.96 \cdot 10^{-130}$
$35 > 10$	$6.67 \cdot 10^{-15}$
greater than $35 > 10$	$6.44 \cdot 10^{-18}$
15 and 20	no difference
$25 > 15$	0.00
$30 > 15$	$2.91 \cdot 10^{-69}$
15 and 35	no difference
greater than $35 > 15$	$6.52 \cdot 10^{-08}$
$25 > 20$	$1.31 \cdot 10^{-06}$
$30 > 20$	$5.12 \cdot 10^{-78}$
$35 > 20$	0.00
greater than $35 > 20$	$7.61 \cdot 10^{-10}$
$30 > 25$	$1.44 \cdot 10^{-27}$
25 and 35	no difference
greater than $35 > 25$	0.00
$30 > 35$	$3.42 \cdot 10^{-14}$
$30 >$ greater than 35	$2.44 \cdot 10^{-06}$
greater than $35 > 35$	0.00

Sensitivities

Methods	**Wilcoxon p-value**
$5 > 10$	0.00
$5 > 15$	$3.16 \cdot 10^{-18}$
$5 > 20$	$1.50 \cdot 10^{-55}$
$5 > 25$	$2.93 \cdot 10^{-26}$
$5 > 30$	$7.16 \cdot 10^{-122}$
$5 > 35$	$3.82 \cdot 10^{-10}$
$5 >$ greater than 35	$3.78 \cdot 10^{-25}$
$10 > 15$	$7.84 \cdot 10^{-07}$
$10 > 20$	$2.70 \cdot 10^{-29}$
$10 > 25$	$6.53 \cdot 10^{-15}$
$10 > 30$	$2.83 \cdot 10^{-70}$
$10 > 35$	$3.80 \cdot 10^{-05}$
$10 >$ greater than 35	$3.20 \cdot 10^{-16}$
$15 > 20$	$5.49 \cdot 10^{-08}$
$15 > 25$	0.00
$15 > 30$	$6.00 \cdot 10^{-30}$
15 and 35	no difference
$15 >$ greater than 35	$4.13 \cdot 10^{-05}$
20 and 25	no difference
$20 > 30$	$3.56 \cdot 10^{-10}$
$35 > 20$	$2.73 \cdot 10^{-06}$
20 and greater than 35	no difference
$25 > 30$	$4.97 \cdot 10^{-12}$
$35 > 25$	0.00
25 and greater than 35	no difference
$35 > 30$	$1.53 \cdot 10^{-17}$
greater than $35 > 30$	$8.59 \cdot 10^{-06}$
$35 >$ greater than 35	$1.31 \cdot 10^{-05}$

Specificities

Methods	Wilcoxon p-value
$10 > 5$	$8.18 \cdot 10^{-07}$
$15 > 5$	0.00
$5 > 20$	$1.61 \cdot 10^{-07}$
$5 > 25$	$2.58 \cdot 10^{-08}$
$5 > 30$	$4.74 \cdot 10^{-10}$
$5 > 35$	0.00
greater than $35 > 5$	$1.34 \cdot 10^{-08}$
10 and 15	no difference
$10 > 20$	$6.24 \cdot 10^{-16}$
$10 > 25$	$8.45 \cdot 10^{-16}$
$10 > 30$	$1.48 \cdot 10^{-19}$
$10 > 35$	$8.82 \cdot 10^{-08}$
greater than $35 > 10$	$1.43 \cdot 10^{-06}$
$15 > 20$	$5.41 \cdot 10^{-13}$
$15 > 25$	$6.40 \cdot 10^{-15}$
$15 > 30$	$3.40 \cdot 10^{-16}$
$15 > 35$	$1.01 \cdot 10^{-07}$
greater than $35 > 15$	$2.69 \cdot 10^{-05}$
$20 > 25$	0.00
20 and 30	no difference
$20 > 35$	0.01
greater than $35 > 20$	$1.15 \cdot 10^{-11}$
25 and 30	no difference
25 and 35	no difference
greater than $35 > 25$	$7.03 \cdot 10^{-12}$
30 and 35	no difference
greater than $35 > 30$	$1.47 \cdot 10^{-12}$
greater than $35 > 35$	$4.83 \cdot 10^{-07}$

"Upper" UAA threshold

CAPS

Precisions

Methods	Wilcoxon p-value
10 > 5	$1.88 \cdot 10^{-05}$
15 > 5	0.01
greater than 15 > 5	0.00
10 and 15	no difference
greater than 15 > 10	0.02
15 and greater than 15	no difference

Sensitivities

Methods	Wilcoxon p-value
5 and 10	no difference
5 and 15	no difference
5 > greater than 15	0.00
10 and 15	no difference
10 > greater than 15	0.01
15 and greater than 15	no difference

Specificities

Methods	Wilcoxon p-value
5 and 10	no difference
5 and 15	no difference
greater than 15 > 5	0.01
10 and 15	no difference
greater than 15 > 10	0.01
15 and greater than 15	no difference

PlotCorr

Precisions

Methods	Wilcoxon p-value
$10 > 5$	$1.25 \cdot 10^{-35}$
$15 > 5$	$4.25 \cdot 10^{-24}$
greater than $15 > 5$	$7.73 \cdot 10^{-35}$
$15 > 10$	$4.21 \cdot 10^{-07}$
greater than $15 > 10$	$2.41 \cdot 10^{-22}$
greater than $15 > 15$	$1.22 \cdot 10^{-11}$

Sensitivities

Methods	Wilcoxon p-value
$5 > 10$	0.00
$5 > 15$	0.02
$5 >$ greater than 15	$2.08 \cdot 10^{-10}$
10 and 15	no difference
$10 >$ greater than 15	0.00
$15 >$ greater than 15	0.00

Specificities

Methods	Wilcoxon p-value
5 and 10	no difference
$15 > 5$	0.02
greater than $15 > 5$	$1.75 \cdot 10^{-21}$
10 and 15	no difference
greater than $15 > 10$	$1.77 \cdot 10^{-13}$
greater than $15 > 15$	$9.80 \cdot 10^{-08}$

MI

Precisions

Methods	Wilcoxon p-value
$10 > 5$	$8.63 \cdot 10^{-40}$
$15 > 5$	$5.49 \cdot 10^{-25}$
greater than $15 > 5$	$9.19 \cdot 10^{-35}$
$15 > 10$	$1.89 \cdot 10^{-10}$
greater than $15 > 10$	$1.34 \cdot 10^{-22}$
greater than $15 > 15$	$1.64 \cdot 10^{-13}$

Sensitivities

Methods	Wilcoxon p-value
5 and 10	no difference
5 and 15	no difference
$5 >$ greater than 15	0.00
10 and 15	no difference
10 and greater than 15	no difference
15 and greater than 15	no difference

Specificities

Methods	Wilcoxon p-value
5 and 10	no difference
5 and 15	no difference
greater than $15 > 5$	$1.61 \cdot 10^{-06}$
10 and 15	no difference
greater than $15 > 10$	0.00
greater than $15 > 15$	0.00

Waddell – Kappa

Precisions

Methods	Wilcoxon p-value
$10 > 5$	$3.78 \cdot 10^{-30}$
$15 > 5$	$4.99 \cdot 10^{-20}$
greater than $15 > 5$	$2.17 \cdot 10^{-34}$
$15 > 10$	0.00
greater than $15 > 10$	$7.52 \cdot 10^{-21}$
greater than $15 > 15$	$5.85 \cdot 10^{-11}$

Sensitivities

Methods	Wilcoxon p-value
$5 > 10$	0.00
$5 > 15$	$5.20 \cdot 10^{-05}$
$5 >$ greater than 15	$1.43 \cdot 10^{-23}$
10 and 15	no difference
$10 >$ greater than 15	$3.27 \cdot 10^{-13}$
$15 >$ greater than 15	$5.44 \cdot 10^{-06}$

Specificities

Methods	Wilcoxon p-value
$10 > 5$	0.02
$15 > 5$	0.01
greater than $15 > 5$	$7.95 \cdot 10^{-22}$
10 and 15	no difference
greater than $15 > 10$	$1.01 \cdot 10^{-12}$
greater than $15 > 15$	$8.04 \cdot 10^{-07}$

Waddell – MI

Precisions

Methods	Wilcoxon p-value
$10 > 5$	$5.97 \cdot 10^{-33}$
$15 > 5$	$3.67 \cdot 10^{-25}$
greater than $15 > 5$	$9.54 \cdot 10^{-35}$
$15 > 10$	$1.16 \cdot 10^{-07}$
greater than $15 > 10$	$8.74 \cdot 10^{-22}$
greater than $15 > 15$	$5.89 \cdot 10^{-10}$

Sensitivities

Methods	Wilcoxon p-value
$5 > 10$	0.00
$5 > 15$	$4.46 \cdot 10^{-05}$
$5 >$ greater than 15	$7.91 \cdot 10^{-34}$
10 and 15	no difference
$10 >$ greater than 15	$3.52 \cdot 10^{-16}$
$15 >$ greater than 15	$3.62 \cdot 10^{-08}$

Specificities

Methods	Wilcoxon p-value
$10 > 5$	$8.30 \cdot 10^{-07}$
$15 > 5$	$7.83 \cdot 10^{-11}$
greater than $15 > 5$	$1.27 \cdot 10^{-27}$
$15 > 10$	$6.07 \cdot 10^{-05}$
greater than $15 > 10$	$1.65 \cdot 10^{-16}$
greater than $15 > 15$	$3.19 \cdot 10^{-06}$

A.10 Combining multiple co-evolution detection methods

We show an example of combining the results of PlotCorr, Waddell – MI, and Waddell – Kappa. Say we are looking at unique amino acid threshold 2, and we consider an alignment with 4 columns (numbering will begin from 0), for which the results are:

- 0 and 1:
  - PlotCorr: -1.86
  - Waddell – MI: 3.80
  - Waddell – Kappa: 8.02
- 0 and 2:
  - PlotCorr: -2.09
  - Waddell – MI: 6.52
  - Waddell – Kappa: -9.19
- 0 and 3:
  - PlotCorr: 8.63
  - Waddell – MI: -4.77
  - Waddell – Kappa: -8.49
- 1 and 2:
  - PlotCorr: -2.42
  - Waddell – MI: 5.56
  - Waddell – Kappa: 0.04
- 1 and 3:
  - PlotCorr: 2.95
  - Waddell – MI: 3.23
  - Waddell – Kappa: 8.06
- 2 and 3:
  - PlotCorr: 0.44
  - Waddell – MI: -5.09
  - Waddell – Kappa: 1.89

Let us say that the truly co-evolving pairs are: (0, 1), (1, 3), and (2, 3). Each of these pairs could be counted as either a true positive, or a false negative. Each of the non-co-evolving pairs (that is, (0, 2), (0, 3), and (1, 2)) could be counted as either a true negative, or a false positive.

For pair (0, 1), we count this as positive (a true positive, in this case) if the following three conditions are met: the PlotCorr score is $>= 2$, the Waddell – MI score is $>= 0$, and the Waddell – Kappa score is $>= 3$. If at least one of these conditions is violated, then the pair is classified is negative (a false negative, in this case). The PlotCorr condition is violated, as its score is -1.86, so this pair is counted as a false positive. Truly negative sites (such as (0,2)) are counted as negative (true negatives) if they do not satisfy at least one of the conditions about the individual scores defined above (i.e. their PlotCorr score is < 2, etc.). They are classed as a false positive if they do satisfy all three conditions.

We obtain the following counts for the above pairs:

- True positives: 1 ((1, 3))
- True negatives: 3 ((0, 2), (0, 3), (1, 2))
- False positives: 0
- False negatives: 2 ((0, 1), (2, 3))

The sensitivity, specificity, and precision statistics can now be calculated as normal.

A.11 Amino acid bias

All of the p-values have been corrected for the number of tests performed (140 = (2 for each co-evolution detection method + conservation) * 20 amino acids), according to a Bonferroni correction. The values in the tables are set to show 2 decimal places, so '0.00' is likely to be a number not small enough to be displayed as $\mathrm{X} \cdot 10^{-\mathrm{Y}}$.

p-values for PlotCorr, with a lower unique amino acid threshold:

Amino acid	**p-value**
A	$6.62 \cdot 10^{-189}$
C	$3.07 \cdot 10^{-120}$
D	0.25
E	$9.43 \cdot 10^{-46}$
F	0.27
G	$2.14 \cdot 10^{-146}$
H	1.00
I	$1.22 \cdot 10^{-201}$
K	0.00
L	0.00
M	1.00
N	0.00
P	$2.63 \cdot 10^{-45}$
Q	0.00
R	$6.69 \cdot 10^{-14}$
S	0.00
T	$7.31 \cdot 10^{-47}$
V	$1.15 \cdot 10^{-292}$
W	$1.30 \cdot 10^{-144}$
Y	0.00

p-values for Waddell – Kappa, with a lower unique amino acid threshold:

Amino acid	**p-value**
A	$3.57 \cdot 10^{-191}$
C	$2.11 \cdot 10^{-175}$
D	1.00
E	$3.45 \cdot 10^{-29}$
F	1.00
G	0.00
H	$6.41 \cdot 10^{-20}$
I	0.00
K	$9.21 \cdot 10^{-15}$
L	0.00
M	0.00
N	$3.97 \cdot 10^{-100}$
P	0.00
Q	$1.07 \cdot 10^{-13}$
R	$6.44 \cdot 10^{-20}$
S	0.00
T	$6.59 \cdot 10^{-168}$
V	0.00
W	0.00
Y	1.00

p-values for Waddell – MI, with a lower unique amino acid threshold:

Amino acid	**p-value**
A	0.00
C	$6.75 \cdot 10^{-228}$
D	0.34
E	$2.05 \cdot 10^{-47}$
F	$1.84 \cdot 10^{-25}$
G	0.00
H	$1.60 \cdot 10^{-44}$
I	0.00
K	0.01
L	$2.80 \cdot 10^{-133}$
M	$5.28 \cdot 10^{-30}$
N	$6.00 \cdot 10^{-185}$
P	0.00
Q	$2.20 \cdot 10^{-37}$
R	$1.28 \cdot 10^{-34}$
S	0.00
T	0.00
V	0.00
W	0.00
Y	$1.02 \cdot 10^{-27}$

p-values for PlotCorr, with a higher unique amino acid threshold:

Amino acid	**p-value**
A	0.00
C	0.04
D	0.09
E	0.00
F	0.00
G	1.00
H	$1.20 \cdot 10^{-13}$
I	0.02
K	1.00
L	$2.11 \cdot 10^{-36}$
M	0.00
N	0.00
P	1.00
Q	1.00
R	1.00
S	1.00
T	$3.16 \cdot 10^{-26}$
V	1.00
W	1.00
Y	0.00

p-values for Waddell – Kappa, with a higher unique amino acid threshold:

Amino acid	**p-value**
A	$2.98 \cdot 10^{-38}$
C	1.00
D	0.00
E	1.00
F	$7.65 \cdot 10^{-49}$
G	0.00
H	$1.16 \cdot 10^{-15}$
I	$1.17 \cdot 10^{-32}$
K	$1.45 \cdot 10^{-14}$
L	$9.34 \cdot 10^{-73}$
M	$8.22 \cdot 10^{-25}$
N	$1.97 \cdot 10^{-20}$
P	0.00
Q	$5.98 \cdot 10^{-30}$
R	1.00
S	1.00
T	$7.68 \cdot 10^{-65}$
V	1.00
W	0.00
Y	$7.90 \cdot 10^{-22}$

p-values for Waddell – MI, with a higher unique amino acid threshold:

Amino acid	**p-value**
A	$4.10 \cdot 10^{-64}$
C	0.00
D	$5.56 \cdot 10^{-40}$
E	1.00
F	$3.70 \cdot 10^{-142}$
G	$5.98 \cdot 10^{-185}$
H	$2.38 \cdot 10^{-177}$
I	$2.08 \cdot 10^{-65}$
K	$2.85 \cdot 10^{-120}$
L	$3.48 \cdot 10^{-294}$
M	$7.95 \cdot 10^{-64}$
N	$1.36 \cdot 10^{-141}$
P	$1.74 \cdot 10^{-275}$
Q	0.00
R	0.00
S	1.00
T	$2.15 \cdot 10^{-261}$
V	0.00
W	$3.72 \cdot 10^{-22}$
Y	$6.43 \cdot 10^{-49}$

p-values for fully conserved sites:

Amino acid	**p-value**
A	0.00
C	0.00
D	$2.78 \cdot 10^{-103}$
E	$6.06 \cdot 10^{-98}$
F	0.00
G	0.00
H	$5.67 \cdot 10^{-111}$
I	0.00
K	0.02
L	0.00
M	0.00
N	0.00
P	0.00
Q	0.00
R	$3.72 \cdot 10^{-79}$
S	0.00
T	0.00
V	0.00
W	0.00
Y	0.00