Visual Search and Analysis in Molecular Biology

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt von
Martin Philipp Heß, M.Sc.
geboren in Fulda.

Erstgutachter:	Prof. Dr.-Ing. Michael Goesele Technische Universität Darmstadt
Zweitgutachter:	Prof. Dr. Martin Weigt Université Pierre et Marie Curie
Drittgutachter:	Prof. Dr. Kay Hamacher Technische Universität Darmstadt

Darmstadt 2017

Heß, Martin Philipp: Visual Search and Analysis in Molecular Biology
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2018
Tag der mündlichen Prüfung: 19.12.2017

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation selbstständig nur mit den angegeben Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 24.10.2017 Martin Philipp Heß

Abstract

The computation of protein sequence alignments is one of the most fundamental tasks in computational biology. Pairwise sequence alignments (PSA) form the basis for the detection of homologous protein sequences. Multiple sequence alignments (MSA) can provide insights into structural and functional relationships across a set of proteins. In the alignment process, evolutionary, functionally, or structurally related regions between the sequences are identified and aligned depending on a particular scoring model. Evolutionary substitution events are usually modeled by substitution matrices, while insertion and deletion events are modeled by specific gap penalties.

The quality of sequence alignments depends heavily on the chosen scoring model, the alignment algorithm, and the sequence data itself. The selection of the best parameters for a given alignment task is, however, non-trivial. Thus many researchers regularly use potentially suboptimal default parameters. This also includes biased and dated substitution matrices. In addition, the construction of MSAs is an NP-complete task and as such the optimal alignment is unknown, even for a fixed parameter set. MSA algorithms thus rely on heuristics to approximate the optimal MSA resulting in alignments of suboptimal quality which often require manual refinement. Assessing the quality of MSAs is also problematic since most established quality measures are limited in the detection of bad alignment regions.

In this thesis, we present several approaches and concepts to improve the accuracy of sequence alignments. In particular, this includes two novel substitution models to enable existing methods to produce better alignments as well as approaches to enable experts and non-experts to assess the quality of the computed MSAs and to effectively refine them to improve their accuracy.

We present the novel CorBLOSUM substitution model that fixes a substantial programming error in the original BLOSUM code. This error negatively affects the homologous sequence search performance of the original BLOSUM matrices as well as their revised RBLOSUM variants. Our exhaustive benchmark analysis based on 51 different ASTRAL subsets shows that CorBLOSUM matrices usually detect more true homologs when compared with their incorrect BLOSUM and RBLOSUM counterparts. For this reason, using CorBLOSUM matrices instead of BLOSUM can substantially improve the results of homologous sequence search.

Furthermore, we propose the novel PFASUM substitution model that is derived from Pfam seed alignments using our novel PFASUM algorithm. Unlike conventional substitution models, our PFASUM matrices are thus based on manually curated expert ground truth data that reflects the currently known sequence space. Additionally, our PFASUM algorithm incorporates several mechanism to avoid oversampling while handling ambiguous amino acids in a reasonable way. As shown by our thorough performance evaluations, these features enable PFASUM matrices to significantly outperform widely used conventional matrices in homologous sequence search. Additionally, using PFASUM matrices for the construction of MSAs also results in more accurate MSAs in most cases.

Beside the aforementioned substitution models, we present a novel visual analysis and comparison approach for protein MSAs. It allows to detect reliably aligned and misaligned regions in protein MSAs without much effort. This is achieved by using an automatic comparison of alternative MSAs of the same sequence set and the visualization of consistently aligned regions and uncertain areas in the MSAs. Our evaluation shows that our system allows to successfully assess the accuracy of MSAs and to effectively determine uncertain regions for further refinement. Additionally, it can be used to visually assess the impact of different alignment algorithms and parameterizations on the resulting alignments.

In order to outsource the cumbersome task of manual MSA refinement, we present our scientific discovery game Bionigma. It abstracts the alignment problem in the form of a puzzle game. In these puzzles, the amino acids in the alignment are represented by different game tokens. Like one would align beads of identical color in an abacus, the players must align similar tokens to improve their score. Through this, the players successively refine the real MSA in a playful manner. Several user studies show that Bionigma is fun to play and delivers a true game experience to the players. Additionally, our results demonstrate that casual players can successfully refine protein MSAs. In particular, they can even produce more accurate than automatic methods.

In summary, the here presented approaches and concepts can help to significantly improve the accuracy of protein sequence alignments. Notably, our methods enable biologists without profound knowledge in the field of sequence alignments to generate better results without much effort.

Zusammenfassung

Die Konstruktion von Proteinsequenzalignments ist eine der fundamentalsten Aufgaben in der Bioinformatik. Paarweise Sequenzalignments (PSA) stellen beispielsweise die Basis für die Detektion homologer Proteinsequenzen. Multiple Sequenzalignments (MSA) erlauben es, Erkenntnisse über strukturelle und funktionale Zusammenhänge zwischen mehreren Protein zu gewinnen. In Abhängigkeit des gewählten Bewertungsmodells werden in einem Sequenzalignment entweder evolutionär, funktional oder strukturell verwandte Sequenzsegmente identifiziert und zueinander angeordnet. Dabei werden evolutionäre Substitutionsereignisse normalerweise durch sogenannte Substitutionsmatrizen modelliert. Insertions- und Deletionsereignisse werden hingegen durch Lücken im Alignment repräsentiert, die mit Strafpunkten bewertet werden.

Die Qualität eines Sequenzalignments hängt maßgeblich vom gewählten Bewertungsschema und Alignmentalgorithmus ab. Zusätzlich haben die Sequenzdaten selbst einen gewissen Einfluss auf die Qualität des Alignments. Da die Selektion dieser Parameter ein schwieriges Problem darstellt, verwenden viele Benutzer suboptimale Standardparameter, wie zum Beispiel ältere oder nachweislich durch Fehler beeinflusste Substitutionsmatrizen. Außerdem stellt die Berechnung von optimalen multiplen Sequenzalignments ein NP-vollständiges Optimierungsproblem dar. Dadurch ist die optimale Alignmentlösung selbst für ein fix gewähltes Parameterset unbekannt. Aus diesem Grund verwenden die meisten Alignmentalgorithmen Heuristiken um das optimale Alignment zu approximieren. Dies resultiert allerdings sehr häufig in suboptimalen Alignments, die manuell weiter verfeinert werden müssen. Leider ist auch die Qualitätsanalyse von MSAs nur sehr eingeschränkt möglich, da existierende Qualitätsmaße tatsächlich schlechte Alignmentregionen nur unzureichend detektieren können.

In dieser Arbeit präsentieren wir daher verschiedene Ansätze, um die Genauigkeit und Qualität von Protein MSAs zu verbessern. Hierfür stellen wir zwei neue Substitutionsmodelle vor, durch deren Verwendung bestehende Alignmentverfahren in der Lage sind, bessere Ergebnisse zu liefern. Zusätzlich stellen wir weitere Verfahren und Ansätze vor, um Experten als auch Nicht-Experten eine bessere Qualitätsanalyse als auch eine effektivere Möglichkeit zur Verfeinerung von MSAs zu bieten.

Hierzu stellen wir das neuartige CorBLOSUM Substitutionsmodell vor. Dieses Modell korrigiert einen Programmierfehler im originalen BLOSUM Programmcode, der sich negativ auf die Fähigkeiten zur homologen Sequenzsuche der BLOSUM als auch der RBLOSUM Matrizen auswirkt. Unsere Ergebnisse basierend auf einer umfassenden Performanzuntersuchung unter der Verwendung von 51 verschiedenen ASTRAL Datenbanken zeigen, dass CorBLOSUM Matrizen in der Regel mehr korrekte homologe Sequenzen detektieren können als die getesteten BLOSUM und RBLOSUM Matrizen. Aus diesem Grund können CorBLOSUM Matrizen substantiell dazu beitragen, bessere Ergebnisse in der homologen Sequenzsuche zu erzielen.

Darüber hinaus präsentieren wir in dieser Arbeit die neuartigen PFASUM Substitutionsmatrizen. Diese Matrizen werden auf Basis von Pfam seed alignments unter Verwendung eines neuartigen Algorithmus erzeugt. Im Gegensatz zu anderen Substitutionsmodellen basieren unsere PFASUM Substitutionsmatrizen somit auf manuell von Experten optimierten Daten, die den aktuell bekannten Proteinsequenzraum abbilden. Außerdem verwendet der PFASUM Algorithmus verschiedene Techniken, um Oversampling zu vermeiden und uneindeutige Aminosäuresymbole in den Alignments geeignet zu verarbeiten. Diese Eigenschaften erlauben es unseren PFASUM Matrizen, konventionelle Substitutionsmodelle in der Suche nach homologen Sequenzen zu übertreffen, wie unsere Studie belegt. Außerdem kann durch die Verwendung unserer PFASUM Matrizen zur Konstruktion von MSAs in den meisten Fällen eine höhere MSA Genauigkeit erzielt werden.

Weiterhin stellen wir einen neuartigen Ansatz zur visuellen Analyse und zum Vergleich von Protein MSAs vor. Mit diesem Ansatz ist es möglich, zuverlässige und falsch angeordnete Regionen in MSAs ohne großen Aufwand zu detektieren. Unser Verfahren nutzt hierzu automatische Vergleichsmaße und verschiedene Visualisierungen, um konsistent angeordnete und unsichere Alignmentregionen hervorzuheben. Unsere Ergebnisse zeigen, dass mit unserem Ansatz die Genauigkeit von MSAs erfolgreich bestimmt werden kann. Außerdem können hierdurch verbesserungswürdige Regionen in den Alignments zur weiteren Optimierung detektiert werden. Unser Ansatz erlaubt es zusätzlich, den Einfluss von verschiedenen Alignmentalgorithmen und Parametrisierungen auf die Struktur der Alignments zu untersuchen.

Um die aufwendige Aufgabe der manuellen Verfeinerung von MSAs auszulagern, stellen wir außerdem einen Computerspielansatz vor. Unser wissenschaftliches Erkundungsspiel Bionigma abstrahiert das Alignmentproblem in Form eines Puzzles. Die Aminosäuren in einem Alignment werden in diesen Puzzlen als Spielsteine repräsentiert. Durch die Anordnung von ähnlichen Spielsteinen, ähnlich dem Anordnen von bunten Perlen in einem Abakus, können die Spieler ihre Punktzahl erhöhen. Hierdurch wird auch das echte Alignment sukzessive verfeinert. Mehrere Benutzerstudien zeigen, dass Bionigma Spaß macht und seinen Spielern eine echte Spielerfahrung bietet. Außerdem zeigen unsere Ergebnisse, das Gelegenheitsspieler erfolgreich MSAs verfeinern und auch unter Umständen MSA Programme übertreffen können.

Zusammenfassend lässt sich sagen, dass die hier vorgestellten Verfahren signifikant zur Verbesserung der Genauigkeit von MSAs beitragen können. Durch unsere Ansätze können Biologen mit weniger profunden Kenntnissen auf dem Gebiet der Sequenzalignments bessere Ergebnisse ohne großen Aufwand erzielen.

Acknowledgements

This thesis would not have been possible without the help of my colleagues, friends, and family. Thus, i would like to thank all these persons for their support and help.

First and foremost, I would like to express my sincere gratitude to my supervisors Prof. Dr.-Ing. Michael Goesele, Prof. Dr. Kay Hamacher, and Prof. Dr. Wiemeyer for their invaluable support and guidance during my time as PhD student. They always supported me with helpful feedback regarding my research, the writing of papers, and this thesis. In particular, I want to thank Prof. Dr.-Ing. Michael Goesele for the opportunity to continue my work at GCC after my initial project was finished. I would also like to thank Prof. Dr. Martin Weigt who kindly agreed to review this thesis.

My special thanks goes to all my colleagues and friends at GCC and CBS for their support, inspiring discussions, and proof-reading of my papers and this thesis. They created a very friendly environment to work in that made my time as PhD student one of the best in my life. In particular, I would like to thank Nicolas for sharing one office with me for the last years. Our interesting discussions and fun times as well as his patience regarding my computer problems made working at GCC a real pleasure. Also, I would like to thank Frank for our inspiring discussions about sequence alignments and fun times at CBS. Without his invaluable support and patience, the realization of our projects would not have been possible. I would also like to thank our student assistants Tobias, Daniel, and Fabian for their support in realizing and implementing Bionigma. Many thanks also go to Ursula for her help and support regarding all administrative aspects of my work.

Furthermore, I would like to thank the forum for interdisciplinary research at TU Darmstadt for funding my first three years as PhD. Additionally, I want to gratefully acknowledge the team of the Hessische Hochleistungsrechner (HHLR) for enabling us to perform some computationally expensive benchmarks.

Finally, I want to express my gratitude to my parents for their patience and support during my studies, and especially to my wife for her continuous support and her understanding during busy times.

Contents

Abstract III

Acknowledgements VII

1 Introduction 1
- 1.1 Biological sequence data . . . 1
- 1.2 Sequence analysis . . . 2
- 1.3 Problem statement . . . 5
- 1.4 Contributions . . . 7
- 1.5 Thesis outline . . . 9

2 Background 11
- 2.1 Scoring models for sequence alignments . . . 11
  - 2.1.1 Substitution events . . . 11
  - 2.1.2 Insertion and deletion events . . . 13
- 2.2 Pairwise sequence alignments . . . 14
  - 2.2.1 Global pairwise sequence alignments . . . 15
  - 2.2.2 Local pairwise sequence alignments . . . 19
- 2.3 Homologous sequence search . . . 21
  - 2.3.1 FASTA . . . 22
  - 2.3.2 BLAST . . . 23
- 2.4 Multiple sequence alignments . . . 26
  - 2.4.1 MSA construction methodologies . . . 27
  - 2.4.2 MSA programs . . . 33
- 2.5 Analysis and comparison of multiple sequence alignments . . . 38
  - 2.5.1 MSA benchmark datasets . . . 39
  - 2.5.2 MSA comparison measures . . . 39

3 Serious games for bioinformatics 41
- 3.1 Introduction . . . 41
- 3.2 Related work . . . 42
  - 3.2.1 Crowdsourcing and citizen science . . . 43
  - 3.2.2 Scientific discovery games in biology . . . 45
- 3.3 First prototype . . . 47
  - 3.3.1 Approach . . . 47
  - 3.3.2 Evaluation . . . 50
  - 3.3.3 Conclusion . . . 53
- 3.4 Second prototype . . . 53
  - 3.4.1 Approach . . . 53
  - 3.4.2 Evaluation . . . 55
  - 3.4.3 Conclusion . . . 57
- 3.5 Bionigma . . . 57
  - 3.5.1 Approach . . . 58

3.5.2 Evaluation - Game experience and usability . . . 63
3.5.3 Evaluation - MSA accuracy . . . 67
3.5.4 Conclusion . . . 71

4 Substitution matrices 75
4.1 Introduction . . . 75
4.2 Related work . . . 76
4.2.1 PAM - Point accepted mutation matrix . . . 76
4.2.2 VTML - Variable time maximum likelihood matrix . . . 77
4.2.3 BLOSUM - Blocks substitution matrix . . . 77
4.2.4 RBLOSUM - Revised BLOSUM matrix . . . 78
4.2.5 Summary . . . 79
4.3 Methods . . . 79
4.3.1 Measuring homology search performance . . . 79
4.3.2 Measuring multiple sequence alignment performance . . . 82
4.4 CorBLOSUM substitution matrices . . . 83
4.4.1 Introduction . . . 83
4.4.2 Algorithm . . . 84
4.4.3 Evaluation . . . 85
4.4.4 Conclusion . . . 99
4.5 PFASUM substitution matrices . . . 101
4.5.1 Introduction . . . 101
4.5.2 Algorithm . . . 102
4.5.3 Evaluation - Homology search performance . . . 106
4.5.4 Evaluation - MSA construction . . . 117
4.5.5 Conclusion . . . 121

5 Visual analysis and comparison of multiple sequence alignments 123
5.1 Introduction . . . 123
5.2 Related work . . . 124
5.2.1 MSA comparison and quality analysis . . . 124
5.2.2 Visual analysis and comparison of MSAs . . . 126
5.3 Approach . . . 129
5.3.1 Quality and comparison measures . . . 131
5.3.2 Visual analysis of multiple sequence alignments . . . 132
5.3.3 Visual comparison of multiple sequence alignments . . . 134
5.4 Evaluation . . . 137
5.4.1 Visual comparison and analysis of HCN MSAs . . . 138
5.4.2 Discussion . . . 141
5.5 Conclusion . . . 142

6 Conclusion 143
6.1 Summary . . . 143
6.2 Discussion . . . 144
6.3 Future work . . . 145

Appendices 148

A Supplemental material - Serious games for bioinformatics **149**

B Supplemental material - CorBLOSUM substitution matrices **153**

C Supplemental material - PFASUM substitution matrices **167**

D (Co-)Authored publications **179**

E Curriculum Vitae **181**

Bibliography **183**

Chapter 1
Introduction

1.1 Biological sequence data

Deoxyribonucleic acid (DNA) and *protein sequences* are essential parts of all living organisms. According to Mount [2004, pp. 5–6], DNA sequences are double-stranded, helical molecules which are composed of a sequence of four different types of *nucleotides*, i.e., adenine (A), guanine (G), cytosine (C), and thymine (T). They form the building blocks of an organism's *genome*, i.e., its entire genetic material (the *genes*) are stored in the chromosomes. *Genes* are functional regions within the DNA that encode "building plans" for the construction of proteins, ribonucleic acid molecules (RNA), and other sequences.

Protein sequences are polypeptides composed of long chains of amino acids [Mount 2004, p. 6]. They are responsible for a vast number of essential functions within the organism. For instance, the well-known protein hemoglobin fulfills the function of oxygen and carbon dioxide transport in human blood [Lesk 2010, p. 76].

The construction process of proteins within living cells using the build plans provided by protein-coding genes is called *protein synthesis* [Lesk 2017, p. 7]. First, the DNA is transcribed into messenger RNA molecules (mRNA). Afterwards, these molecules travel to the ribosome where they are used as a template for the construction of a protein strand. Each triplet of nucleotides in an mRNA sequence (so-called *codons*) is translated into a particular amino acid in the protein. The type of an amino acid, usually denoted as an one letter code, is defined by the genetic code represented by the codon (Table 1.1).

In most organisms, the codons encode twenty different types of amino acids which are denoted as *standard amino acids* or *canonic amino acids* (Table 1.1). However, certain organisms also produce additional amino acids such as selenocysteine (U) and pyrrolysine (O) (Table 1.1). As described by Lesk [2010, p. 371], these non-standard amino acids are synthesized by reinterpreting stop codons.

As outlined above, proteins (or peptides) fulfill a vast amount of functions within the organisms. According to Lesk [2010, pp. 2, 68–80], this includes, e.g., controlling DNA replication, processing stimuli responses, enzyme catalysis, and transporting different kinds of molecules. There are also structural proteins such as the keratins found in hair and nails of mammals, antibody proteins that repel invading pathogens, and transducer proteins that convert chemical to mechanical energy. The function of a particular protein is determined by its folding in three-dimensional space which in turn is defined by the *amino-acid residues* in the protein sequence, their order and biochemical properties.

According to the *IUPAC Compendium of Chemical Terminology* [McNaught and Wilkinson 1997], an "amino-acid residue" refers to what remains of each amino acid in a peptide when the elements of water are removed. In the context of protein sequences, this term is usually abbreviated with "residue" and denotes a specific amino acid inside the protein.

Amino acid	Three letter code	One letter code	Codons
Alanine	Ala	A	GCU, GCC, GCA, GCG
Arginine	Arg	R	CGU, CGC, CGA, CGG, AGA, AGG
Asparagine	Asn	N	AAU, AAC
Aspartic acid	Asp	D	GAU, GAC
Cysteine	Cys	C	UGU, UGC
Glutamic acid	Glu	E	GAA, GAG
Glutamine	Gln	Q	CAA, CAG
Glycine	Gly	G	GGU, GGC, GGA, GGG
Histidine	His	H	CAU, CAC
Isoleucine	Ile	I	AUU, AUC, AUA
Leucine	Leu	L	CUU, CUC, CUA, CUG, UUA, UUG
Lysine	Lys	K	AAA, AAG
Methionine	Met	M	AUG
Phenylalanine	Phe	F	UUU, UUC
Proline	Pro	P	CCU, CCC, CCA, CCG
Serine	Ser	S	AGU, AGC, UCU, UCC, UCA, UCG
Threonine	Thr	T	ACU, ACC, ACA, ACG
Tryptophan	Trp	W	UGG
Tyrosine	Tyr	Y	UAU, UAC
Valine	Val	V	GUU, GUC, GUA, GUG
Selenocysteine	Sec	U	UCA
Pyrrolysine	Pyl	O	UAG

Table 1.1: Table of standard and non-standard amino acids and their corresponding one letter codes, three letter codes, and their codons.

1.2 Sequence analysis

Since the DNA of an individual organism is unique, the amino acid composition of specific protein sequences obtained from different organisms usually differs to some extent. The detection and analysis of these differences as well as the similarities between a set of protein sequences can reveal precious information that is useful for several important biological tasks and applications. It allows, e.g., to identify active sites or core regions in the proteins (domains), to gather information about the evolutionary process (point mutations and conserved regions), or to conclude on similar structure and function (3D structure prediction). The obtained insights form the basis for further crucial applications such as the development of new drugs for medical treatment and the inference of evolutionary relationships among different species (phylogeny). Differences and similarities between a set of sequences are usually detected and examined by so-called sequence analysis methods. This includes in particular *pairwise sequence alignments* (PSA), *multiple sequence alignments* (MSA), and *homologous sequence search*.

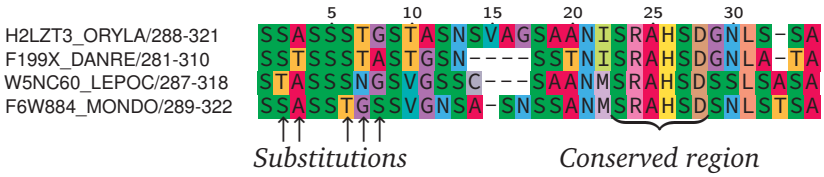


Figure 1.1: Exemplary (multiple) sequence alignment of four protein segments obtained from the Pfam seed alignment of family PF15814 [Finn et al. 2016]. Different amino acid types are depicted by differently colored boxes and their one letter code representation. Single or consecutive dash symbols represent gaps referring to evolutionary insertion or deletion events. Columns containing mismatching amino acid types (e.g., those marked with arrows) indicate substitution events, while columns with similar or identical amino acid types reflect evolutionary stable regions (marker on the right hand side).

Sequence alignments

In a *sequence alignment*, either evolutionary, functionally, or structurally related segments (homologous regions) between a set of protein sequences are aligned by inserting gaps of different length into the sequences. This is similar to the process of aligning identically colored beads in an abacus [Chatzou et al. 2016]. Sequence alignments are typically separated into two different classes, *pairwise sequence alignments* (PSA) containing only two sequences and *multiple sequence alignments* (MSA) aligning more than two sequences. Both classes have different application purposes and are described later in this section. Notably, in the context of this thesis we refer to alignments as protein sequence alignments based on evolutionary relationships.

Figure 1.1 shows an exemplary alignment of four protein sequences with column numbers shown at the top. Each row in the alignment represents a single sequence. The corresponding names and sequence numbers are shown on the left hand side. Individual amino acid types within the sequences are depicted by differently colored boxes and labeled with their corresponding one letter codes. Gaps introduced to form the alignment are highlighted by consecutive dash symbols on white background.

Evolutionary events and scoring models

Aligned amino acids and gaps in a sequence alignment indicate different evolutionary events (Section 2.1). Columns containing amino acids of identical type imply evolutionary stable or conserved regions (e.g., columns 23 to 28 in Figure 1.1). In contrast, columns containing mismatching amino acids indicate evolutionary substitution events caused by point mutations in the DNA (e.g., marked columns on the left hand side of Figure 1.1). The mismatching threonine (T) residue in the second column, e.g., either indicates a substitution of serine (S) with threonine (T) in sequence `W5NC60_LEPOC/287-318` or the inverse substitution of threonine (T) with serine (S) in the other three sequences. The exact “direction” of this substitution cannot be determined from the alignment alone.

Consecutive gap symbols in the alignment indicate evolutionary **in**sertion or **del**etions events, so-called indels. An insertion event represents the effect that a chain of amino acids is inserted into a particular sequence during evolution. In contrast, a deletion event denotes the opposite effect that a particular chain of amino acids was removed from a sequence during the evolutionary process. Which kind of indel event a particular gap represents, often cannot be directly determined from the alignment. A single amino acid in a column otherwise exclusively containing gap symbols, e.g., is likely to be caused by an insertion event. Distinguishing insertion and deletion events in a column containing the same number of gaps and amino acids is, however, not possible.

Formally, the construction of a sequence alignment corresponds to an optimization problem given a particular objective function that defines which sequence segments are related and thus should be aligned or not. For example, the alignment process is often modeled as the process of maximizing the residue similarity within the columns according to a specific similarity measure while simultaneously minimizing the number of gaps.

This is typically achieved by applying penalties for the introduction of gaps and rating pairs of aligned amino acids by a particular similarity matrix (Section 2.1.2). In the case of protein alignments, so-called substitution matrices such as PAM [Dayhoff et al. 1978], (R/Cor)BLOSUM [Henikoff and Henikoff 1992a, Styczynski et al. 2008, Hess et al. 2016a], VTML [Müller and Vingron 2000, Müller et al. 2002], or PFASUM [Keul et al. 2017] are used to rate pairs of aligned amino acids. These matrices represent the relative mutation rates between two amino acid types α_i and α_j in the form of (rounded) log-odds scores. In Chapter 4, we thoroughly describe and discuss different substitution matrices and their application purposes in detail.

Pairwise sequence alignments

Pairwise sequence alignments under a particular scoring model can be computed either locally or globally (Section 2.2). Local PSAs only align specific segments of two proteins that are strongly similar, while global PSAs represent full end-to-end alignments of the sequences. While the former enables the identification of similar regions in otherwise dissimilar sequences, the latter allows to compare two sequences and to judge their similarity, e.g., by counting the number of aligned amino acids of identical type. The PSA shown in Figure 1.2, e.g., reveals 20 pairs of identical amino acids between the two sequences (blue). Since the sequence shown at the top has a length of 34 amino acids, its similarity to the shorter sequence at the bottom corresponds to $20/34 \approx 59\%$.

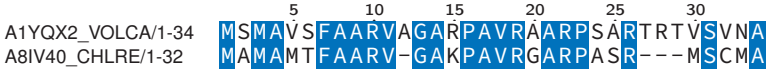


Figure 1.2: Pairwise alignment of the sequence segments `A1YQX2_VOLCA/1-34` and `A8IV40_CHLRE/1-32` obtained from the Pfam seed alignment of family PF11591 [Finn et al. 2016]. Pairs of identical amino acids are highlighted in blue.

PSAs form the basis for several applications in computational biology such as homologous sequence search, i.e., the detection of evolutionary related protein sequences (homologs) [Mount 2004, p. 71]. For instance, homology search tools (BLAST [Altschul et al. 1990], PSI-BLAST [Altschul et al. 1997] and FASTA [Pearson 1991]) usually identify homologs by comparing a query sequence with the sequences in a large database based on pairwise sequence alignments. Searching for homologs is one of the most common tasks in computational biology, e.g., when analyzing newly determined sequences [Pearson 2013]: Whenever a new protein is discovered, identifying homologs may allow to transfer valuable details from these homologs to the newly discovered protein such as structural and functional properties. More detailed information about homologous sequence search and corresponding programs are provided in Section 2.3.

Multiple sequence alignments

While the analysis of protein sequences with PSAs allows to transfer information about a single sequence to another, the comparison of more than two sequences by so-called *multiple sequence alignments* (MSA) (Figure 1.1) can yield additional and potentially more profound information about a set of proteins (Section 2.4). For this reason, MSAs are the basis for various biological applications and research areas. According to Chatzou et al. [2016], this includes, e.g., domain analysis, phylogenetic reconstruction, and motif finding. Other applications are the analysis of co-evolutionary effects and the detection of key functional residues. For example, if a newly determined protein is associated with a specific disease, multiple sequence alignment of this protein and homologous sequences may reveal similar regions that are known to be responsible for the proper functioning of the protein. This information can then be used to develop drugs affecting these key functional residues in order to interfere with the protein's function.

Analogous to pairwise alignments, MSAs align evolutionary, functionally, or structurally related regions between sequences by introducing gaps (Figure 1.1). The computation of an optimal global MSA, however, necessitates the simultaneous consideration of the relationships between multiple sequences which is way more complex than only aligning two sequences. In fact, the construction of an MSA has been proven to be an NP-complete optimization problem [Wang and Jiang 1994, Just 2001, Elias 2006]. For this reason, MSA algorithms typically rely on some kind of heuristic in order to approximate the optimal MSA under a given scoring model. We provide a detailed description of different MSA construction methodologies and an overview of a broad range of MSA programs in Section 2.4.

1.3 Problem statement

As outlined in the previous section, pairwise and multiple sequence alignments form the basis for a vast amount of important tasks such homologous sequence search, phylogeny and drug design. Thus, the success of these applications and the reliability of their results strongly depends on the quality of the generated alignments. If a homologous sequence search program computes incorrect pairwise alignments, e.g., the resulting list of homologs probably contains a large number of false positives. This may result in misclassified protein families which in turn results in inaccurate phylogenetic trees and so on.

In general, the quality of protein sequence alignments depends on three main factors: the scoring model, the alignment algorithm, and the sequence data. These main factors are discussed in more detail in the following paragraphs.

Impact of the evolutionary scoring model

An evolutionary scoring model for the construction of sequence alignments usually involves at least a particular substitution matrix for modeling evolutionary substitution events and a specific gap penalty model for encoding evolutionary insertion and deletion events. Sometimes further criteria are considered such as the physico-chemical properties of the amino acids [Katoh et al. 2002] or additional secondary structure information [Wright 2015].

The selection of the "best" scoring model for a particular alignment task is generally a non-trivial problem [Giribet and Wheeler 1999, Reese and Pearson 2002, Price et al. 2005, Agrawal and Huang 2009, Hess et al. 2014a]. For this reason, commonly used default scoring models are often employed for sequence alignments which are usually quite dated and may be suboptimal for a given alignment task. Commonly used substitution matrices are typically derived by counting amino acid pairs in automatically aligned sequence datasets. While this method itself provides a plausible way to infer substitution events, it obviously requires correctly aligned datasets to produce reliable results. This is, however, a chicken-and-egg problem since aligning a sequence dataset for inferring substitution events in turn also requires a scoring model. A possible solution to break this loop is the creation and usage of manually aligned sequence datasets. However, this requires domain specific expert knowledge and is a very time consuming process.

In addition, widely used substitution matrices such as the popular BLOSUM [Henikoff and Henikoff 1992a] or PAM matrices [Dayhoff et al. 1978] are derived from dated and quite small datasets (Chapter 4). As shown by previous studies (e.g., Price et al. [2005]) as well as our own evaluations presented in Chapter 4, this may limit their capabilities for the alignment of sequences when considering that the known sequence space available today provides a much larger basis for the construction of substitution matrices.

Besides the substitution matrix, the chosen gap penalties have another huge impact on the resulting alignments. If the penalty values are too high, the number of gaps is strongly limited resulting in an over-alignment. On the other hand, low penalties may introduce too many gaps which results in an under-alignment. In general, the heights of the gap penalties should depend on the chosen substitution matrix.

Impact of the alignment algorithm

The alignment algorithm chosen for a particular task has a large impact on the resulting alignment. Optimal pairwise alignment of two sequences under a given scoring model can be computed using algorithms such as Needleman-Wunsch [Needleman and Wunsch 1970] or Smith-Waterman [Smith and Waterman 1981]. In contrast, the computation of MSAs under the typically used sum-of-pairs scoring scheme (Section 2.4.1) is an NP-complete optimization problem [Wang and Jiang 1994, Just 2001, Elias 2006].

MSA algorithms such as `MUSCLE` [Edgar 2004b] or `MAFFT` [Katoh et al. 2002] therefore rely on heuristics to approximate the optimal MSA. This often leads to local misalignments resulting in MSAs of suboptimal quality which requires cumbersome manual refinement.

In addition, the assessment of an MSA's quality is non-trivial since the optimal MSA for a particular scoring model is unknown. Hence, quality measures are required to judge the local and global quality of an alignment. While state-of-the-art quality measures may reveal probably correctly aligned regions, they cannot necessarily make reliable assertions about bad alignment regions (Chapter 5). Another problem in this context is the lack of visual tools for the quality assessment and comparison of MSAs. For these reasons, the cumbersome process of manually refining MSAs usually requires expert domain knowledge.

Impact of the underlying sequence data

Besides the aforementioned problems, the sequence data itself has a strong impact on the alignment process. The alignment of very similar sequences is usually much easier than the computation of alignments only containing distantly related sequences. An explanation for this behavior can be derived from the diagonal values of most substitution matrices. Typically, the diagonal entries describing mutation rates between identical amino acid types are much larger than the off-diagonal entries. In other words, the alignment of identical amino acid pairs is usually favored over substitution events. Since similar sequences are typically closely related and thus contain large numbers of conserved amino acid regions, their alignment contains fewer "questionable" decisions induced by similar SP scores. The alignment of distantly related sequences therefore requires a scoring model which contains enough information about substitution events covering a sufficient evolutionary time frame in order to identify the distant relationships.

Summary

In summary, there are a number of problems that arise in the context of sequence alignments which have a huge impact on their quality. This includes in particular the properties of the available substitution matrices, the need of expert knowledge for the manual refinement of MSAs, and the lack of tools and reliable measures for the quality analysis of MSAs. In this thesis, we address these issues by several contributions to the research field of sequence alignments. A detailed list of these contributions is provided in the following section.

1.4 Contributions

In this thesis, we present several contributions to the research field of protein sequence alignments and its applications such as homologous sequence search and the computation of multiple sequence alignments. This includes the presentation of novel substitution matrices for improved homology search results and MSA quality, a citizen science approach for manual MSA refinement and an improved technique for the visual comparison and quality assessment of MSAs.

The main contributions have been previously published as papers at international peer-reviewed conferences and journals. These papers are presented in this thesis partially in verbatim. They are combined with additional results in the form of separate sections structured in three main parts. Each main part refers to a specific research field in the context of protein sequence alignments: manual MSA refinement, substitution matrices, and MSA quality analysis. The contributions are as follows:

Part I - Serious games for bioinformatics

- In this part, we present an improved citizen science game approach for the manual refinement of protein multiple sequence alignments. Our approach abstracts the alignment problem as a puzzle game. This enables non-experts to successfully improve sequence alignments while having fun and experiencing a true game experience. Several user studies and the quality scores obtained for the resulting MSAs based on state-of-the-art quality measures provide evidence for our claims.

  Chapter 3, [Hess et al. 2014b]

  This citizen science game approach was developed by myself. This work was supervised by Prof. Dr. Wiemeyer, Prof. Dr. Kay Hamacher, and Prof. Dr.-Ing. Michael Goesele.

Part II - Substitution matrices

- We first present the novel CorBLOSUM substitution matrix series [Hess et al. 2016a]. The CorBLOSUM algorithm removes an additional programming error in the clustering routine of the original BLOSUM code [Henikoff and Henikoff 1992b] which affects the original BLOSUM [Henikoff and Henikoff 1992a] and RBLOSUM matrices [Styczynski et al. 2008]. As shown by our evaluation, the usage of these matrices for the task of homologous sequence search can lead to significantly improved search results.

  Chapter 4 - Section 4.4, [Hess et al. 2016a]

  The CorBLOSUM matrix series was jointly developed by Frank Keul and myself. This work was supervised by Prof. Dr.-Ing. Michael Goesele and Prof. Dr. Kay Hamacher.

- Second, we present a novel matrix series called PFASUM. This matrix represents a novel evolutionary scoring model based on manually curated structural alignments that cover the currently known sequence space. Our thorough evaluation shows that the usage of PFASUM matrices for homology search produces significantly better search results than commonly used conventional substitution matrices. Additionally, our results demonstrate that employing PFASUM matrices for MSA construction can substantially improve the quality of the resulting MSAs.

  Chapter 4 - Section 4.5, [Keul et al. 2017]

  Similar to the CorBLOSUM matrix series, the PFASUM matrix series was jointly developed by Frank Keul and myself. This work was also supervised by Prof. Dr.-Ing. Michael Goesele and Prof. Dr. Kay Hamacher.

Part III - Visual analysis and comparison of multiple sequence alignments

- In the third part of this thesis, we propose a visual analytics approach for the visual interactive comparison and quality assessment of multiple sequence alignments. This approach allows the visual interactive detection and exploration of potentially correct

as well as misaligned MSA regions. This technique supports the user in judging the quality of an MSA and allows the manual refinement of the MSA for improved quality.

Chapter 5, [Hess et al. 2016b]

This visual analysis and comparison approach was developed by myself. This work was supervised by Prof. Dr. Wiemeyer, Prof. Dr. Kay Hamacher, and Prof. Dr.-Ing. Michael Goesele.

1.5 Thesis outline

Chapter 2: Background

In Chapter 2, we introduce the reader to the area of protein sequence alignments and its applications in computational biology. First, we explain the different event types that lead to changes in protein sequences during evolution and how these events are usually measured. In the second part, we introduce the concept of pairwise sequence alignments and explain how these alignments can be constructed using state-of-the-art algorithms and their purpose within the context of homologous sequence search. In the third and last part, we introduce multiple sequence alignments and discuss their purpose and how they are constructed by state-of-the-art algorithms.

Chapter 3: Serious games for bioinformatics

Chapter 3 describes the concepts of citizen science and serious games. It presents Bionigma, our citizen science game approach for improving protein MSAs. First, we discuss existing crowdsourcing and citizen science game approaches and in particular those that address problems from the biology domain. In the second part, we present the different developmental stages of our citizen science game approach for the manual refinement of protein. The remainder of this chapter presents the results of our user studies evaluating Bionigma's capability to provide game experience to the players and the quality of the obtained MSA refinements.

Chapter 4: Substitution matrices

In Chapter 4, we discuss commonly used protein substitution matrices for homologous sequence search and MSA computation and present two novel matrix series which lead to improved homology search and MSA results. The CorBLOSUM matrix series corrects for a substantial error that can be found in BLOSUM and RBLOSUM matrices which results in improved homology search performance. The second matrix series presented in this thesis is called PFASUM. It is derived from manually curated structural alignments using a novel algorithm. Hence, PFASUM matrices represent scoring models based on the currently known protein sequence space. As the results demonstrate, the usage of this novel matrix series leads to significantly better homologous sequence search results and can substantially improve the quality of MSAs.

Chapter 5: Visual analysis and comparison of multiple sequence alignments

Chapter 5 presents an improved visual analytics approach for the visual comparison and quality assessment of MSAs. First, we provide the reader with a short motivation about this particular topic by discussing existing approaches for the visual exploration of MSAs and commonly used MSA quality measures. The remainder of this chapter presents our own approach for the interactive visual comparison and quality analysis of MSAs on the basis of existing quality and comparison measures as well as our own insights obtained during the development of Bionigma.

Chapter 6: Conclusion

In this last chapter, we conclude this thesis by providing a summary of the presented contributions and a discussion of the proposed approaches. We also highlight different aspects that could be improved by future research.

Chapter 2
Background

2.1 Scoring models for sequence alignments

The fundamental concept behind sequence analysis methods such as homologous sequence search is the construction of sequence alignments. The construction of a sequence alignment is the process of aligning either evolutionary, functionally, or structurally related sequence segments by inserting gaps of different lengths into the sequences. This is similar to one would align colored beads in an abacus. Notably, gaps can also be added to the ends of the sequences to align their terminal regions with non-terminal areas in other sequences. These gaps are denoted as terminal gaps. Solving this optimization problem requires specific scoring models to define which sequence segments are related or represent parts that were inserted or deleted during evolution (gaps). In particular, this includes modelling evolutionary substitution, insertion, and deletion events (Figure 2.1).

2.1.1 Substitution events

In the context of protein sequences, evolutionary substitution events describe the effect that amino acids in the sequence are substituted with different acid types during evolution. This effect is initiated by point mutations in the DNA which alter the encoded amino acid type used for the protein synthesis. In protein sequence alignments, substitution events are indicated by mismatching amino acid pairs (red highlighted regions in Figure 2.1). In contrast, aligned amino acids of identical type either indicate no change during evolution or point mutations in the DNA not affecting the encoded acid type (bold blue letters in Figure 2.1). For instance, DNA point mutations at the last nucleotide of the base triplet ACT do not have an impact on the encoded amino acid type since the four possible resulting mRNA triplets ACU, ACG, ACA, and ACC encode the same amino acid threonine (T) (Table 1.1).

Figure 2.1: Pairwise sequence alignment of the sequence segments *A1YQX2_VOLCA/1-34* and *A8IV40_CHLRE/1-32* obtained from the Pfam seed alignment of family *2Fe-2S_Ferredox* (PF11591, Pfam release 31.0) [Finn et al. 2016]. Preserved amino acids are shown in bold blue letters, while mismatching amino acid pairs indicating substitutions are highlighted in red. Alignment regions referring to evolutionary insertions or deletions are highlighted in orange.

	A	R	G	M	S	T	V
A	5	-1	0	-1	1	0	0
R	-1	7	-2	-2	-1	-1	-3
G	0	-2	8	-4	0	-2	-4
M	-1	-2	-4	8	-2	-1	1
S	1	-1	0	-2	5	2	-2
T	0	-1	-2	-1	2	6	0
V	0	-3	-4	1	-2	0	5

Table 2.1: Selection of PFASUM60 log-odds scores encoding the relative mutation rates between the amino acid types alanine (A), arginine (R), glycine (G), methionine (M), serine (S), threonine (T), and valine (V).

Substitution events between amino acids are usually modeled by substitution matrices. For all pairs of two amino acid types α_i and α_j, these matrices represent the likelihood that type α_i mutates into α_j in relation to independent evolution. The likelihood of preserving an amino acid type, i.e., $\alpha_i = \alpha_j$, is represented on the diagonal. These likelihoods, i.e., relative mutation rates, are typically encoded as log-odds scores. By representing the likelihoods in the log-domain, the joint probability of multiple substitutions (e.g., within a PSA) can simply be calculated by summing over the log-odds scores instead of computing the product of the underlying likelihoods [Lesk 2017, p. 168]. Additionally, they are often rounded for numerical reasons.

The relative mutation rates represented by a substitution matrix are usually derived from observed amino acid pairs in aligned sequence datasets with known sequence relationships. From these alignments it is, however, not possible to determine if an observed amino acid pair AC refers to the substitution of A with C or vice versa [Lesk 2017, p. 185]. Hence, substitution matrices are usually symmetric. An exemplary substitution matrix is depicted in Table 2.1. The log-odds scores for the substitution events between the amino acids alanine (A), arginine (R), glycine (G), methionine (M), serine (S), threonine (T), and valine (V) are obtained from our novel PFASUM60 substitution matrix (Section 4.5).

If one applies this scoring model to the pairwise sequence alignment shown in Figure 2.1, e.g., the first amino acid pair in the alignment (MM) yields a score of +8 for preserving methionine. In contrast, the substitution of a serine residue (S) with an alanine residue (A) at the second column is only rated with +1. As indicated by the different scores, preserving methionine residues (MM) is considered to be more likely than substitutions between serine and alanine (SA) and vice versa (AS). This is in concordance with common knowledge that preserving amino acid types during evolution is more likely than amino acid substitutions. Most substitution matrices thus possess much larger scores on the diagonal than on the off-diagonal. Amino acid substitutions also tend to be conservative, i.e., substitutions between amino acids of comparable size or similar physicochemical properties are considered to be more likely than between amino acids with larger differences [Lesk 2013, p. 184]. The PFASUM60 matrix, e.g., scores substitutions between the similar amino acid types valine (V) and methionine (M) by +1, while substitutions between the dissimilar amino acids valine (V) and arginine (R) are considered to be less likely and thus receive negative scores (−3).

Notably, this section only provides a short introduction about substitution matrices in order to prepare the reader for the following sections. More detailed information about substitution matrices, e.g., different matrix series, their application purposes, and construction methodologies, is presented later in Chapter 4.

2.1.2 Insertion and deletion events

Evolutionary insertion or deletion events (indel) describe the effect that an amino acid or a chain of amino acids are inserted or deleted in a protein sequence during the evolutionary process. Similar to amino acid substitutions, this effect is also initiated by point mutations in the DNA. In sequence alignments, indels are represented in the form of gaps which are usually denoted by consecutive dash (-) or dot (.) symbols (orange regions in Figure 2.1).

Gaps of a specific length l are typically rated using some sort of penalty model. The simplest model for rating gaps is the *constant gap penalty model* which penalizes each gap with a fixed value regardless of its length. In contrast, the *linear gap penalty model* $g(l) = l \cdot \omega$ additionally takes the gap length into account by penalizing each gap symbol in a contiguous gap with a fixed value ω. For instance, a gap of length $l = 5$ would receive a penalty five times as large as a gap of length one.

From a biological perspective, both methods have some severe disadvantages. The constant gap penalty does not take the length of the gap into account which can result in unrealistically long gaps. In turn, the linear gap penalty model artificially reduces the number of longer gaps by applying very high penalties. The linear gap penalty model thus contradicts common knowledge that long contiguous gaps are considered to occur more likely than the same number of indels at noncontiguous positions [Lesk 2013, p. 184] since they can be caused by single mutations in the DNA [Gotoh 1982].

To overcome these limitations, one of the most commonly used gap penalty models is the *affine gap penalty model* shown in Equation 2.1:

$$g(l) = \omega + (l-1) \cdot \epsilon \tag{2.1}$$

It can be considered as a hybrid of the constant and linear gap penalty model. A gap of length l is penalized based on two different penalties – a gap opening or existence penalty ω and a gap extension penalty ϵ. While the opening cost ω penalizes each gap by a fixed amount analogous to the constant penalty model, the extension cost ϵ linearly penalizes the gap depending on its length.

The height of the opening penalty is usually chosen larger than the extension penalty. This accounts for common knowledge that indel events are considered to occur less likely during evolution than amino acid substitutions, but also mitigates the costs for longer gaps. For example, when using an affine penalty model with a gap opening penalty of $\omega = -10$ and a gap extension penalty of $\epsilon = -1$, a gap of length $l = 5$ yields a penalty of $g(5) = -10 + (5-1) \cdot (-1) = -14$.

When rating sequence alignments, the initial penalties are usually adapted depending on the values of the chosen substitution matrix. There is also a consensus that gaps are more likely to occur at specific positions in the alignment. Terminal gaps, i.e., those that occur at the ends of the sequences, are often less penalized in order to cope with sequences of different lengths [Thompson et al. 1994, Edgar 2004b]. Likewise, gaps aligned with consecutive

stretches of hydrophilic residues are considered to be more likely than those aligned with hydrophobic amino acids and are thus often penalized with reduced costs [Thompson et al. 1994, Edgar 2004b]. Some alignment algorithms also employ residue-specific gap penalties depending on the type of the amino acids that are aligned with gaps [Thompson et al. 1994] or profile-based gap penalty functions [Katoh et al. 2002]. The latter assigns variable gap penalties in dependency of position-specific gap distributions observed for an alignment profile.

The selection of the optimal gap penalty model for a given alignment task is typically a non-trivial problem [Giribet and Wheeler 1999]. It depends on various factors, e.g., the chosen substitution matrix as mentioned before, and the similarity of the sequences that should be aligned. Evaluating gap penalty models and choosing biologically meaningful gap parameters is thus still part of active research [Benner et al. 1993, Giribet and Wheeler 1999, Reese and Pearson 2002, Young and Healy 2003, Agrawal and Huang 2009, Eddy 2009, Wang et al. 2011].

2.2 Pairwise sequence alignments

As outlined in the introduction of this thesis, pairwise sequence alignments form the basis for several applications and research tasks in computational biology. They are often used to identify homologous sequences [Altschul et al. 1990, Pearson and Lipman 1988] or to infer structural and functional information of an unknown protein from a related sequence with known properties [Mount 2004, p. 71].

The construction of a PSA corresponds to the task of aligning homologous regions between sequences by inserting gaps according to a particular scoring model, i.e., a substitution matrix and a gap penalty model. More formally, the computation of a PSA represents an optimization problem which aims at maximizing a specific similarity score. In contrast to the construction of multiple sequence alignments, which is an NP-complete optimization problem (Section 2.4), optimal PSAs under a given scoring model can be calculated using dynamic programming (DP). The main difference between PSA algorithms [Needleman and Wunsch 1970, Hirschberg 1975, Smith and Waterman 1981, Gotoh 1982] are their runtime complexity, memory consumption, and the resulting PSA type, i.e., global or local PSAs.

A global pairwise sequence alignment represents a full end-to-end alignment of both sequences A and B, i.e., all amino acids of A and B are aligned to each other. However, if the lengths of A and B differ to a greater extend or if A and B are very dissimilar, a large number of gaps has to be inserted in order to form a global PSA. Hence, global PSAs are usually generated for homologous sequences that are quite similar and of approximately equal length [Mount 2004, p. 70]. In contrast, local pairwise sequence alignments only align local regions of high similarity between two sequences A and B. This allows, e.g., to detect homologous regions in otherwise non-homologous or dissimilar sequences. Local PSAs are thus often used when searching for potentially homologous sequences, e.g., when employing tools such as BLAST [Altschul et al. 1990] or SSEARCH [Pearson 1991]. In the following subsections, we describe different DP algorithms for the calculation of global and local PSAs in detail.

2.2.1 Global pairwise sequence alignments

Needleman-Wunsch algorithm

The Needleman-Wunsch algorithm [Needleman and Wunsch 1970] is the most popular DP algorithm for the computation of a global pairwise sequence alignment. In the initial publication from 1970, the Needleman-Wunsch algorithm was only informally described. The formal definition of the algorithm in the form of a matrix recurrence relation presented here, is based on the publication "*Some Biological Sequence Metrics*" by Waterman et al. [1976].

In order to calculate the global PSA of two sequences A and B of lengths m and n, first an $(n+1) \times (m+1)$-matrix M is initialized according to the following equations:

$$\begin{aligned} M(0,0) &= 0 \\ M(i,0) &= M(i-1,0) + g(i) \\ M(0,j) &= M(0,j-1) + g(j) \end{aligned} \tag{2.2}$$

The variables i and j are indices referring to the uninitialized matrix entries ranging between $1 \leq i \leq m$ and $1 \leq j \leq n$, respectively. The function $g(x)$ represents the cost for inserting a gap of length x with $x = i \vee x = j$. Afterwards, the remaining matrix entries are recursively calculated as defined by the following matrix recurrence relation:

$$M(i,j) = max \left\{ \begin{array}{c} M(i-1,j-1) + S(\alpha_i, \alpha_j) \\ max_{1 \leq k \leq i} \{M(i-k,j) + g(k)\} \\ max_{1 \leq l \leq i} \{M(i,j-l) + g(l)\} \end{array} \right\} \quad \begin{array}{ll} \text{Substitution} & \nwarrow \\ \text{Deletion} & \uparrow \\ \text{Insertion} & \leftarrow \end{array} \tag{2.3}$$

$S(\alpha_i, \alpha_j)$ represents the log-odds score for the pairing of the i-th residue of sequence A with the j-th residue of sequence B. Here, α_i and α_j denote the type of the corresponding residues. The value of each entry in M represents the score of the optimal alignment of the sequence prefixes $A[1..i]$ and $B[1..j]$. Hence, the score of the full global alignment of A and B corresponds to the matrix entry $M(n,m)$, i.e., the similarity of A and B. While this only yields the score of the global PSA of A and B, the alignment itself can be obtained by storing the decisions made in the construction process (Equation 2.3, labels on the right hand side) in a separate matrix $\widehat{M}$ and backtracking. Notably, there can be multiple optimal alignments that yield the same maximum score, but their editing path obtained through backtracking is different.

The Needleman-Wunsch algorithm has a runtime complexity of $O(max(n,m)^3)$ for the calculation of a PSA of two arbitrary sequences A and B of lengths n and m using linear gap costs $g(x)$. Its space complexity corresponds to $O(nm)$ for storing the $(n+1) \times (m+1)$-matrices M and $\widehat{M}$. In 1975, Hirschberg [1975] presented a divide-and-conquer strategy to reduce the space complexity of the Needleman-Wunsch algorithm. Hirschberg's algorithm only requires linear space in dependency of the sequence length but comes at the cost of doubled runtime [Löytynoja and Goldman 2005].

When choosing a fixed gap penalty cost ω, the Needleman-Wunsch algorithm can be specialized in order to compute the optimal global PSA in only $O(n^2)$. We refer to this specialized form of the Needleman-Wunsch algorithm in concordance to common practice as the *standard DP algorithm* for the computation of optimal global pairwise sequence alignments.

Standard DP algorithm

Instead of searching the maximum scores for the deletion and insertion events within a given row or column index range (Equation 2.3), the *standard DP algorithm* only requires the calculation of the maximum of the neighboring cells as shown in Equation 2.4:

$$M(i,j) = max \left\{ \begin{array}{c} M(i-1,j-1) + S(\alpha_i, \alpha_j) \\ M(i-1,j) + \omega \\ M(i,j-1) + \omega \end{array} \right\} \quad \begin{array}{cc} \text{Substitution} & \nwarrow \\ \text{Deletion} & \uparrow \\ \text{Insertion} & \leftarrow \end{array} \tag{2.4}$$

The matrix is identically initialized as in Equation 2.2 with the exception that the linear gap function $g(x)$ is replaced by the fixed gap penalty ω.

The following example (Table 2.2 and Table 2.3) illustrates the standard DP algorithm and the backtracking procedure for the PSA of two sequence segments A and B (*A1YQX2_VOLCA/12-17* and *A8IV40_CHLRE/12-16*) obtained from the Pfam seed alignment [Finn et al. 2016] of family *2Fe-2S_Ferredox* (PF11591, Pfam release 31.0). For this example, a fixed gap penalty of $\omega = -1$ is used in combination with our novel PFASUM60 substitution matrix (Table 2.1, Section 4.5).

Table 2.2 shows the matrices M and $\widehat{M}$ after the initialization step (Equations 2.2). In order to differentiate the three evolutionary events in the backtracking matrix $\widehat{M}$, we use different arrow symbols. A substitution is denoted by a diagonal arrow ($\nwarrow$) in the backtracking matrix $\widehat{M}$. Insertions and deletions are indicated by left arrows ($\leftarrow$) and upper arrows ($\uparrow$), respectively.

Afterwards, the missing entries are calculated either row- or column-wise starting at $M(1,1)$ which results in the matrices shown in Table 2.3. For instance, this entry corresponds to the maximum alignment score of the segments A[1..1] and B[1..1] obtained for three different evolutionary events. For a substitution event, the score equates to 7, i.e., the sum of the diagonal predecessor $M(0,0) = 0$ and the log-odds score $S(\alpha_1, \alpha_1) = S(R,R) = 7$ of the residue pair ($A[1] = R$, $B[1] = R$). The score for an insertion event is the maximum of the score of the vertical predecessor $M(i-1,j)$ plus the fixed gap penalty of $\omega = -1$. Thus, the cost for an insertion at $M(1,1)$ is -2. Similarly, the score for a deletion event corresponds to the score of the horizontal predecessor $M(i,j-1)$ plus the fixed gap penalty. This also yields a score of -2 for a deletion event at $M(1,1)$. The score for the alignment of A[1..1] and B[1..1] is then $M(1,1) = max\{7,-2,-2\} = 7$. Since this score is achieved by a substitution event, a diagonal arrow ($\nwarrow$) is inserted into the backtracking matrix at $\widehat{M}(1,1)$ in order to reconstruct the editing path of the final alignment.

As mentioned above, the score of the optimal PSA of two sequences A and B with sequence lengths n and m is stored in the matrix entry $M(n,m)$. In our example, this corresponds to the entry $M(5,4) = 24$ shown in red on the left hand side of Table 2.3. In order to reconstruct the corresponding alignment, we backtrack the decisions made during the construction of M using the backtracking matrix $\widehat{M}$. Starting at $\widehat{M}(n,m) = \widehat{M}(5,4) = \nwarrow$, we simply follow the arrows until $\widehat{M}(0,0)$ is reached. On the right hand side of Table 2.3, the resulting path is highlighted by red arrows. For each $\nwarrow$, we align the residues of both sequences at the current index. Similarly, a gap is introduced at the current index into A or B when observing an $\leftarrow$ or $\uparrow$ arrow. As noted above, there may be more than one editing path resulting in

	-	R	V	A	G	A
-	0	-1	-2	-3	-4	-5
R	-1					
V	-2					
G	-3					
A	-4					

	-	R	V	A	G	A
-	↖	←	←	←	←	←
R	↑					
V	↑					
G	↑					
A	↑					

Table 2.2: The Needleman-Wunsch matrix M (left) and the backtracking matrix $\widehat{M}$ (right) after the initialization step using a fixed gap penalty of $\omega = -1$. In the backtracking matrix, a diagonal arrow (↖) indicates a substitution, a left arrow (←) an insertion and an upper arrow (↑) a deletion.

	-	R	V	A	G	A
-	0	-1	-2	-3	-4	-5
R	-1	7	6	5	4	3
V	-2	6	12	11	10	9
G	-3	5	11	12	19	18
A	-4	4	10	16	18	**24**

	-	R	V	A	G	A
-	↖	←	←	←	←	←
R	↑	↖	←	←	←	←
V	↑	↑	↖	←	←	←
G	↑	↑	↑	↖	↖	←
A	↑	↑	↑	↖	↑	↖

Table 2.3: Result of the Needleman-Wunsch algorithm for the pairwise alignment of the sequence segment *A1YQX2_VOLCA/12-17* with the segment *A8IV40_CHLRE/12-16* using our novel PFASUM60 matrix (Table 2.1, Section 4.5) and a fixed gap penalty of $\omega = -1$. Both segments were obtained from the Pfam seed alignment of family *2Fe-2S_Ferredox* (PF11591, Pfam release 31.0). The matrix on the left hand side depicts the scoring matrix M of the algorithm with the final alignment score shown in red. The matrix on the right hand side shows the backtracking matrix $\widehat{M}$. A diagonal arrow (↖) indicates a substitution, a left arrow (←) an insertion in *A8IV40_CHLRE*, and an upper arrow (↑) a deletion in *A1YQX2_VOLCA*. The arrows highlighted in red mark the editing path from the lower right to the upper left corner that leads to the final pairwise alignment.

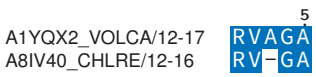


Figure 2.2: Optimal global pairwise sequence alignment for the sequence segments *A1YQX2_VOLCA/12-17* and *A8IV40_CHLRE/12-16* based on the standard DP algorithm using a fixed gap penalty of $\omega = -1$ in combination with the PFASUM60 substitution matrix (Table 2.1, Section 4.5).

multiple optimal global PSAs solutions under a given scoring model. In the shown example exists, however, only one valid editing path. Its corresponding optimal global PSA is shown in Figure 2.2.

Gotoh algorithm

Another algorithm for the calculation of the optimal pairwise alignment of two sequences A and B of length n and m is the Gotoh algorithm [Gotoh 1982], a modified version of the Needleman-Wunsch algorithm. In contrast to the original algorithm, Gotoh's modification has a reduced runtime complexity of $O(nm)$ and uses an affine gap penalty model (Equation 2.1).

In order to cope with the affine gap penalty model, the formalization of Gotoh's algorithm as matrix recurrence relations requires three matrices of size $(m+1)\cdot(n+1)$. These matrices M, P, and Q are initialized as follows:

$$\begin{aligned} M(0,0) &= P(0,0) = Q(0,0) = 0 \\ M(i,0) &= Q(i,0) = M(0,j) = P(0,j) = -\infty \\ P(i,0) &= g(i) \\ Q(0,j) &= g(j) \end{aligned} \tag{2.5}$$

The variables i and j are index variables referring to the uninitialized matrix entries ranging between $1 \leq i \leq m$ and $1 \leq j \leq n$, respectively. These entries are calculated using the following matrix recurrence relations in combination with the aforementioned gap penalty parameters and a scoring matrix S:

$$\begin{aligned} M(i,j) &= max \left\{ \begin{matrix} M(i-1,j-1) + S(i-1,j-1) \\ P(i-1,j-1) + S(i-1,j-1) \\ Q(i-1,j-1) + S(i-1,j-1) \end{matrix} \right\} \\ P(i,j) &= max \left\{ \begin{matrix} M(i-1,j) + \omega \\ P(i-1,j) + \epsilon \\ Q(i-1,j) + \omega \end{matrix} \right\} \\ Q(i,j) &= max \left\{ \begin{matrix} M(i,j-1) + \omega \\ P(i,j-1) + \omega \\ Q(i,j-1) + \epsilon \end{matrix} \right\} \end{aligned} \tag{2.6}$$

Analogous to the Needleman-Wunsch algorithm, the entries of the matrices M, P, and Q represent the score of the optimal alignment of particular prefixes of the sequences A and B. Here, $M(i,j)$ corresponds to the alignment score of the prefixes of $A[1..i]$ and $B[1..j]$ under affine gap costs, while $P(i,j)$ and $Q(i,j)$ represent the score of the same prefixes that end with a gap in A or gap in B, respectively. Again, the final pairwise alignment can be obtained by storing the decisions made during the construction of M, P, and Q using three matrices $\widehat{M}$, $\widehat{P}$, and $\widehat{Q}$ and employing backtracking. In this case, however, the backtracking algorithm runs over three matrices instead of one.

2.2.2 Local pairwise sequence alignments

Smith-Waterman algorithm

As outlined in Section 2.2, local pairwise alignments are another important form of pairwise sequence alignments, e.g., to identify homologous regions in dissimilar sequences. Instead of fully aligning two sequences, a local PSA only aligns strongly similar regions between a query sequence A and a target sequence B. The most prominent algorithm for the calculation of local PSAs is the Smith-Waterman algorithm [Smith and Waterman 1981].

This algorithm also represents a specialization of the aforementioned Needleman-Wunsch algorithm using an affine gap penalty $g(l) = \omega + (l-1) \cdot \epsilon$. Instead of initializing the first row and column of M with the linear gap penalties of a specific length and maximizing over three cases in the matrix recurrence relation, the Smith-Waterman algorithm initializes the first row and column of M with zero (Equation 2.7) and maximizes over four cases (Equation 2.8).

$$\begin{aligned} M(i,0) &= 0 \\ M(0,j) &= 0 \end{aligned} \tag{2.7}$$

$$M(i,j) = max \left\{ \begin{array}{c} 0 \\ M(i-1,j-1) + S(\alpha_i, \alpha_j) \\ M(i-1,j) + g(i) \\ M(i,j-1) + g(j) \end{array} \right\} \quad \begin{array}{cc} \text{Empty suffix} & \odot \\ \text{Substitution} & \nwarrow \\ \text{Deletion} & \uparrow \\ \text{Insertion} & \leftarrow \end{array} \tag{2.8}$$

In contrast to the Needleman-Wunsch algorithm, which holds the score of the optimal PSA of sequences A and B in the matrix entry $M(n,m)$ with n and m being the lengths of A and B, the score for the optimal local PSA can be found elsewhere in matrix M. It corresponds to the largest value $max(M)$ in M with the indices x_1 and y_1. Likewise, the second largest value in M at index $M(x_2, y_2)$ represents the score of the second best local alignment. Again, the final optimal local PSA itself can be obtained by storing the decisions made during the construction of M in a matrix $\widehat{M}$ and backtracking. However, this traceback starts at $\widehat{M}(x_1, y_1)$ and stops as soon as an entry in $\widehat{M}$ is reached that refers to the empty suffix. Analogous to global PSAs, there can be multiple optimal solutions for local PSAs resulting in identical scores but different tracebacks and thus alignments.

The following example (Table 2.4 and Table 2.5) illustrates the functionality of the Smith-Waterman algorithm for the construction of the optimal local PSA of the sequence segments *A1YQX2_VOLCA/26-30* and *A8IV40_CHLRE/23-26* obtained from the Pfam seed alignment of family *2Fe-2S_Ferredox* (PF11591, Pfam release 31.0). The PFASUM60 matrix (Table 2.1, Section 4.5) is used for rating substitution events, while gaps are penalized using a gap opening penalty of $\omega = -10$ and a gap extension penalty of $\epsilon = -1$. Table 2.4 shows the matrices M and $\widehat{M}$ after the initialization step (Equation 2.7). The final matrix entries based on the matrix recurrence relation defined in Equation 2.8 are shown on the left hand side of Table 2.5. Here, the highest local alignment score $max(M) = 9$ can be observed at the entry $M(3,3)$ highlighted in red. The decisions made for each entry are stored in the backtracking matrix $\widehat{M}$ (right). A diagonal arrow ($\nwarrow$) indicates a substitution and a $\odot$ symbol is used for the empty suffix. The red arrows show the editing path for obtaining the optimal local PSA according to the alignment score of $M(3,3) = 9$. This alignment is shown in Figure 2.3.

	-	R	T	R	T	V
-	0	0	0	0	0	0
A	0					
S	0					
R	0					
M	0					

	-	R	T	R	T	V
-	⊙	⊙	⊙	⊙	⊙	⊙
A	⊙					
S	⊙					
R	⊙					
M	⊙					

Table 2.4: The Smith-Waterman matrix M (left) and the backtracking matrix $\widehat{M}$ (right) after the initialization step. A ⊙ symbol in the backtracking matrix indicates the empty suffix.

	-	R	T	R	T	V
-	0	0	0	0	0	0
A	0	0	0	0	0	0
S	0	0	2	0	2	0
R	0	7	0	**9**	0	0
M	0	0	6	0	8	1

	-	R	T	R	T	V
-	⊙	⊙	⊙	⊙	⊙	⊙
A	⊙	⊙	⊙,↖	⊙	⊙,↖	⊙,↖
S	⊙	⊙	↖	⊙	↖	⊙
R	⊙	↖	⊙	↖	⊙	⊙
M	⊙	⊙	↖	⊙	↖	↖

Table 2.5: Result of the Smith-Waterman algorithm for the pairwise alignment of the sequence segment *A1YQX2_VOLCA/26-30* with the segment *A8IV40_CHLRE/23-26* using our novel PFASUM60 matrix (Table 2.1, Section 4.5) in combination with a gap opening penalty of $\omega = -10$ and a gap extension penalty of $\epsilon = -1$. Both segments were obtained from the Pfam seed alignment of family *2Fe-2S_Ferredox* (PF11591, Pfam release 31.0). The matrix on the left hand side depicts the scoring matrix M of the algorithm, the right hand side shows the backtracking matrix $\widehat{M}$. An ↖ arrow indicates a substitution and a ⊙ symbol the empty suffix. The arrows highlighted in red mark the editing path for the construction of the optimal local pairwise alignment.

```
A1YQX2_VOLCA/26-30  TR
A8IV40_CHLRE/23-26  SR
```

Figure 2.3: Optimal local pairwise sequence alignment for the sequence segments *A1YQX2_VOLCA/26-30* and *A8IV40_CHLRE/23-26* based on the Smith-Waterman algorithm. The PFASUM60 matrix (Table 2.1, Section 4.5) was chosen as scoring model in combination with a gap opening penalty of $\omega = -10$ and a gap extension penalty of $\epsilon = -1$.

2.3 Homologous sequence search

Searching large protein databases for homologs of a particular query sequence is one of the most common tasks in computational biology. For example, it is one of the first and most informative steps when analyzing newly determined sequences [Pearson 2013] since it potentially allows to transfer valuable knowledge about known sequences to the newly discovered protein such as structural and functional properties.

Commonly used homology search programs (FASTA [Pearson and Lipman 1988], SSEARCH [Pearson 1991], BLAST [Altschul et al. 1990], and PSIBLAST [Altschul et al. 1997] identify potential homologs based on local pairwise sequence alignments, i.e., based on relative sequence similarity. The PSAs are constructed during the search process by aligning each protein sequence in the database to the query sequence typically using variations of the Smith-Waterman algorithm (FASTA and BLAST). As a rule of thumb, PSAs with "high" SP scores usually indicate similar sequences which are often considered to be homologous and thus may represent valid search hits to the given query sequence. A list of potential homologs to the query sequence can then be obtained by sorting the list of target sequences in descending order of their PSA SP scores.

The disadvantage of this method is that it does not provide any information about the significance of the search results. This is related to the fact that there is no definition at which magnitude the SP score of a PSA indicates a true homologous sequence relationship or not. Since the SP score is based on amino acid pairs, the score obviously scales with the total number of possible amino acid pairs in the sequence dataset which depends on the number of sequences and their lengths. Likewise, the differences between the SP scores of a number of PSAs only reveal which PSA contains more similar sequences compared to other PSAs, but the differences do not properly reflect the degree of diversity between the sequences in terms of a distance metric.

For this reason, a so-called *E-value* or *expect value* is typically calculated for each computed alignment which measures the significance of a search result based on statistics often empirically obtained for a particular scoring model and sequence lengths:

$$E = Kmn\,\mathrm{e}^{-\lambda S} \tag{2.9}$$

Here, m and n refer to the sequence lengths in the PSA and S to the obtained alignment score. The parameters K and λ characterize the statistics of the alignment scores under a particular scoring model and sequence lengths. They are usually estimated empirically by analyzing the scores of a sufficient number of optimal local PSAs of either unrelated sequences [Collins et al. 1988, Pearson 1998] or those obtained from a random sequence model [Altschul and Gish 1996].

In simple terms, the *E-value* represents the number of unrelated sequences that achieve a local alignment score as least as high as the score obtained for the computed PSA of the query and matching sequence [Mount 2004]. The *E*-value thus provides a statistical measure for the identification of a reported search result as a true homolog purely by chance. Homology search results for a given query sequence are typically presented in ascending order of their *E*-values instead of their alignment scores.

Lesk [2013, p. 195] provides rough guidelines for interpreting *E*-values. According to these guidelines, a search result with an *E*-value of $E \leq 0.02$ probably represents a true homologous sequence to a query sequence, while *E*-values between 0.02 and 1 indicate unproven homology but also cannot be ruled out. In contrast, *E*-values of $E > 1$ represent search results probably found purely by chance. Notably, the author states that statistics may provide a useful guide but cannot substitute careful thinking about the results. Hence, manual inspection of the obtained search results by an expert is essential to retrieve reliable information.

In the following subsections, we describe and discuss the methodology of the two most commonly used packages for searching protein databases for homologous sequences given a particular query sequence — the FASTA [Pearson 1991] and the BLAST [Altschul et al. 1990] packages.

2.3.1 FASTA

One of the first software packages for rapidly searching DNA or protein sequences in large sequence databases is the *FASTA package* by Pearson and Lipman [1988]. This package provides two different programs for the identification of homologs given a particular protein sequence as query and a protein database as target, namely the programs SSEARCH and FASTA. SSEARCH performs full optimal local PSAs using the Smith-Waterman algorithm (Section 2.2.2), while FASTA employs heuristics to speed up the search process. For this reason, SSEARCH typically delivers more accurate search results than FASTA but at the cost of increased computation time.

The following description of the FASTA algorithm is based on the book by Mount [2004, pp. 240–247] and the publication by Pearson and Lipman [1988]. FASTA performs four steps in order to compute the pairwise similarity score of two sequences. The intermediate results of these steps are shown in Figure 2.4. Each subfigure illustrates the solution space for a local PSA of a query sequence and a particular database sequence at a given step. The shown diagonals indicate possible local alignments.

First, the ten best-matching regions between the query sequence and each database sequence are identified. According to Mount [2004, p. 242], this is done by rapidly locating words of length k (k-tuples) in both sequences that share the same position offsets by using a lookup table. Matching k-tuples within a certain distance d to each other are then merged including their intermediate regions to form joint matches. In the context of protein sequences, values of either $k = 1$ and $d = 32$ or $k = 2$ and $d = 16$ are chosen. From these joint matches, again the ten best matching regions are selected for further processing (red lines in Figure 2.4 a).

In the second step, the best-matching regions are rescored using a user-specified scoring matrix. For each best-matching region, a subregion with maximal score, the so-called *initial region*, is determined (red highlights in Figure 2.4 b). Subsequently, FASTA identifies "compatible" initial regions that can be joined to create a single alignment (blue diagonals in Figure 2.4 c). This is done by calculating an optimal alignment of the initial regions based on their locations and scores, and a specific *joining penalty*. Low-scoring initial regions (red diagonals in Figure 2.4 c) are not taken into account during this step in order to limit degradation of selectivity. The score of the resulting alignment is then used to rank the corresponding database sequence for further processing.

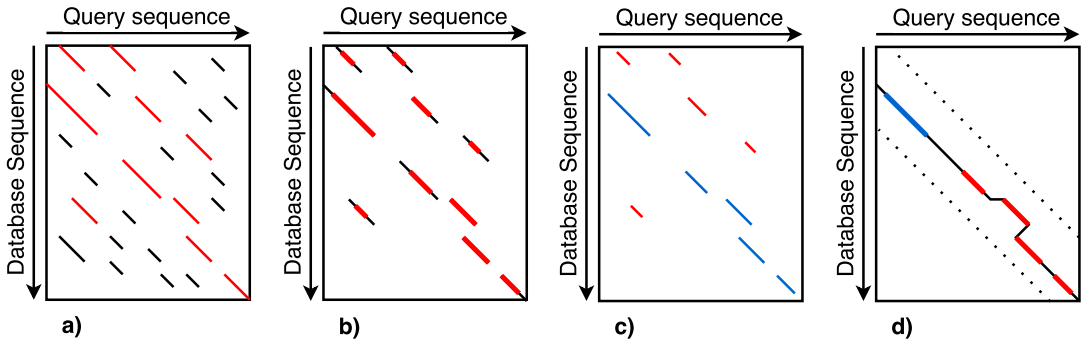


Figure 2.4: Example illustrating the results of the four different steps executed by FASTA to compute the similarity score between a query and a database sequence. a) Identified matching regions (joined *k*-tuples) with the ten best matching regions highlighted in red. b) Identification of initial regions (red) within the ten best matching regions based on substitution scores. c) Identification of "compatible" regions (blue) that can be used to form a single alignment. Red diagonals depict low-scoring initial regions that are omitted by applying a scoring threshold. d) The final pairwise sequence alignment. Red diagonals represent the initial regions from the previous step. The highest scoring initial region that limits the solution space for the final alignment (dotted band) is highlighted in blue.

In the last step, the sequences with the highest rank are optimally aligned to the query sequence using a modified version of the Smith-Waterman algorithm. Instead of taking all possible alignments into account, this algorithm only considers those solutions that lie within a certain band (dotted lines Figure 2.4 d) around the highest scoring initial region (blue diagonal, Figure 2.4 d).

FASTA and SSEARCH report the significance of the search results by presenting *E*-values. In both programs, the *E*-values are calculated based on an extreme value distribution from the mean and variance of the local alignment scores obtained for the PSAs of unrelated sequences [Pearson 1998].

2.3.2 BLAST

The *Basic Local Alignment Search Tool* (BLAST) [Altschul et al. 1990] is the most widely used software for homologous sequence search today [Pearson 2013]. Similarly to the previously described FASTA package, BLAST provides several programs for comparing DNA or protein sequences with protein or DNA databases in any combination. For instance, protein databases can be searched with BLAST using the programs blastp [Altschul et al. 1997], PSI-BLAST [Altschul et al. 1997], PHI-BLAST [Zhang et al. 1998], and DELTA-BLAST [Boratyn et al. 2012].

From this collection, blastp represents the initial BLAST program for homologous sequence search. PSI-BLAST is a further development of blastp which produces significantly better search results by employing so-called position-specific substitution matrices (PSSM) at the cost of higher runtime. A more specialized kind of search can be performed with PHI-BLAST. It limits the search results to contain only those sequences that match a user-specified input pattern. The last development of the BLAST package is DELTA-BLAST. It works similar to

PSI-BLAST but uses PSSMs constructed with information retrieved from a search in the Conserved Domain Database. In the following, we describe the most commonly used BLAST programs for homologous sequence search — blastp and PSI-BLAST — in more detail.

BLASTP

The standard BLAST program for the identification of homologous protein sequences is blastp. The following description of the blastp algorithm is based on the book *Bioinformatics: Sequence and Genome Analysis* by Mount [2004]. Similar to FASTA, the speed of the alignment process is increased by searching small matching patterns (k-tuples) in the query sequence and the database sequences. However, blastp only takes the most significant k-tuples into account instead of searching for all possible words of length k. This effectively reduces the number of k-tuples that need to be processed which provides blastp with a speed advantage over FASTA. However, blastp uses preset word sizes of $k \in \{2, 3, 6\}$ as default[1] and thus may provide less sensitivity than FASTA which compares words of size $k \in \{1, 2\}$.

First, blastp identifies all words of length k within the query sequence, i.e., the k-tuples K_i with i denoting the index of a tuple. For all of these k-tuples K_i, blastp determines all j possible k-tuples $M_{i,j}$ of which the ungapped alignment to K_i would yield a score of at least T given a particular substitution matrix (e.g., BLOSUM62). The following example illustrates this step. Assuming an alphabet of 20 different amino acid types and query sequence tuples K_i of the length $k = 3$, there are $20 \times 20 \times 20 = 8000$ possible 3-tuples that could be aligned without gaps to each K_i. Many of these 8000 tuples yield, however, bad alignment scores. This indicates that these are very unlikely to occur in related sequences. For instance, the alignment of the 3-tuple K_i = PQG with itself (the perfect match) yields the highest BLOSUM62 score of $7 + 5 + 6 = 18$. In contrast, the very dissimilar 3-tuple FCL results in a negative BLOSUM62 score of $-4 - 3 - 4 = -11$. By applying a scoring based threshold filtering, blastp only considers k-tuples with similar scores to the perfect match, i.e., those j k-tuples $M_{i,j}$ that are likely to be found in homologous sequences. For $k = 3$ and a scoring threshold $T = 13$, e.g., this effectively reduces the number of tuples that has to be searched for each single K_i from 8000 to approximately 50.

Afterwards, all database sequences are searched rapidly for exact matches to any of the j most significant k-tuples $M_{i,j}$ for each tuple K_i in the query sequence. Exact matches are joined if they lie within a certain distance to each other on the same diagonal in the alignment space. They are then extended in both directions until their score does no longer increase.

Next, the significance of the scores of the extended sequence stretches, the so-called high-scoring segment pairs (HSP), is determined by calculating E-values for each obtained score. The parameters K and λ applied in the E-value equation (Equation 2.9) are chosen depending on the substitution matrix used for scoring the HSPs. Blastp provides predefined parameters for calculating E-values for each available substitution matrix that were empirically derived from scores obtained for ungapped alignments of random sequences. The calculated E-values are then used to determine which HSPs are sufficiently significant to produce a local sequence alignment.

[1] Blastp preset word sizes were obtained from the NCBI protein-protein BLAST web service available at https://blast.ncbi.nlm.nih.gov (last accessed 15.09.2017)

If more than one significant HSP is identified, `blastp` computes a local alignment and determines the significance of the resulting alignment score in the form of *E*-values using parameters K and λ based on gapped alignment statistics. These parameters are also empirically determined via an analysis of the score distribution of alignments of random sequences. These alignments were constructed using the same scoring matrix but also using identical gap penalties.

Finally, the sequences containing significant HSP *E*-values are locally aligned to the query sequence using the Smith-Waterman algorithm. For each of the obtained alignments, an *E*-value is computed using statistical parameters for gapped alignments. These *E*-values are then used to rank the corresponding sequences in the list of search results.

PSI-BLAST

According to Lesk [2013, pp. 200–201], `blastp` allows quite accurate and rapid detection of homologs to a particular query under the assumption that the query and the target sequences are relatively similar. However, more distantly related homologs cannot be detected accurately by `blastp` in many cases. `PSI-BLAST` addresses this sensitivity problem by employing so-called *position-specific substitution matrices* (PSSM). As reported by Lesk [2013, p. 201], this allows `PSI-BLAST` to find three times as many correct homologs than `blastp` in the so-called *twilight zone* of sequence identities below 30%.

Instead of encoding the relative substitution rate of an amino acid in relation to another specific amino acid type (pairs of amino acids), position-specific substitution matrix scores encode the likelihood of observing a particular amino acid type at a specific position in an alignment (Table 2.6). This information is typically derived from multiple sequence alignments by calculating the relative frequencies of each amino acid type separately for each alignment column.

In the following, we describe the work flow of `PSI-BLAST` for the detection of homologous protein sequences according to the information provided by Lesk [2013, pp. 200–201]. An overview of the `PSI-BLAST` algorithm is shown in Figure 2.5. In the first step of the algorithm, `PSI-BLAST` performs a standard gap enabled `blastp` search as previously explained using a user-specific scoring model, i.e., a substitution matrix and affine gap penalties. The obtained list of search results is then filtered using an *E*-value threshold to only keep the most significant hits that are very likely to represent true homologous sequences. According to Lesk [2013, p. 200], e.g., a frequently chosen *E*-value threshold is 0.005. In the next step, a

	A	**R**	**N**	**D**	**C**	⋯
Pos. 1	-1	-9	-3	3	2	⋯
Pos. 2	-2	4	-2	-2	-1	⋯
Pos. 3	-5	4	-2	2	-2	⋯
Pos. 4	-4	-1	-6	-3	0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 2.6: Example illustrating a position specific scoring matrix (PSSM). The log-odds score for the likelihood of alanine (A) occurring at the first position in an alignment is -1, while the probability of observing an aspartic acid (D) at the third position is $+2$.

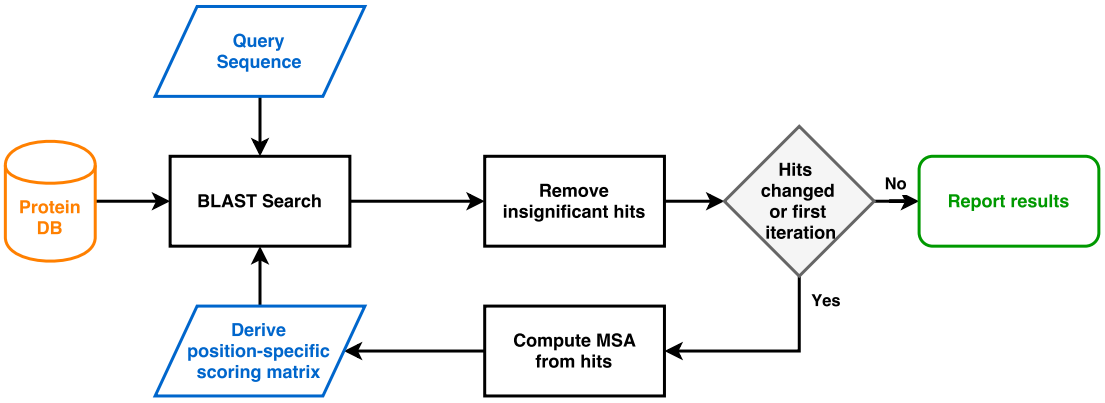


Figure 2.5: Flowchart illustrating the program sequence when searching a protein database for homologs to a given query sequence using PSI-BLAST. First, a gap enabled blastp search is performed using a standard substitution matrix (e.g., BLOSUM62) which yields an initial list of search hits. Second, the list is filtered to remove insignificant results based on a certain *E*-value threshold. In the third step, the remaining sequences are multiply aligned to the query sequence and the resulting MSA is used to derive a position-specific scoring matrix. Afterwards, a new BLAST search is triggered using the position-specific scoring matrix instead of the initially used standard substitution matrix. This procedure is repeated until the obtained list of search hits does no longer change significantly.

multiple sequence alignment is computed for the remaining sequences. Based on this MSA, `PSI-BLAST` counts the relative frequencies of the amino acids with respect to their position in the alignment in order to derive a position-specific substitution matrix. The obtained PSSM can then be used as a scoring matrix for a further `blastp` search which may identify additional homologs. This process can be repeated multiple times with the PSSM derived in iteration i being used as the substitution model in iteration $i+1$. According to Lesk [2013, p. 201], this loop is interrupted as soon as a single iteration yields only little or no change in the obtained search hits.

2.4 Multiple sequence alignments

While pairwise sequence alignments primarily allow to measure the similarity of two sequences, multiple sequence alignments (Figure 2.6) can provide much more information. They can be used, e.g., for the construction of Hidden Markov Models (HMM) which enable categorization of protein sequences into different families [Finn et al. 2016]. Other research areas that depend on MSAs include domain analysis and phylogenetic reconstruction [Chatzou et al. 2016].

Similar to pairwise sequence alignments, the construction of an MSA under a particular scoring model is the process of either aligning evolutionary, functionally, or structurally related regions between sequences by inserting gaps of different lengths. However, the computation of an MSA necessitates the simultaneous consideration of the relationships between multiple sequences which is way more complex than only aligning two sequences. For example, inserting gaps into any of the sequences shown in Figure 2.6 would affect all pairwise induced alignments of this particular sequence to all other sequences and thus the entire MSA. Hence, an MSA algorithm has to maximize the similarity score of all pairwise

Figure 2.6: Multiple sequence alignment of four sequence segments obtained from the Pfam seed alignment of family *FAM199X* (PF15814, Pfam release 31.0). Shown are the segments *H2LZT3_ORYLA/61-385*, *F199X_DANRE/56-374*, *W5NC60_LEPOC/62-381*, and *F6W884_MONDO/67-385* in the column interval $[220, 260]$. Conserved amino acid regions are indicated by bold blue letters. Columns with mismatching amino acid pairs indicating point mutations in the DNA are shown in red. Orange columns highlight indel regions in the shown MSA.

induced alignments in order to compute a globally optimal alignment. Unfortunately, this has been proven to be an NP-complete optimization problem [Wang and Jiang 1994, Just 2001, Elias 2006]. For this reason, MSA algorithms typically rely on some kind of heuristic in order to approximate the optimal MSA under a given scoring model.

In the following subsections, we provide a broad overview of different methodologies and programs for computing protein multiple sequence alignments. First, we introduce the reader into the most widely used scoring measure for MSA construction, the so-called sum-of-pairs score (SP), and explain how the standard DP approach can be used to calculate the optimal MSA. Subsequently, we describe different methodologies that are commonly employed in the construction process of MSA programs. The remainder of this section, presents and describes a representative selection of state-of-the-art MSA programs and algorithms for constructing protein MSAs.

2.4.1 MSA construction methodologies

Sum-of-pairs score

As outlined above, the computation of a global multiple sequence alignment necessitates the simultaneous optimization of the scores of all pairwise induced sequence alignments encoded in the MSA. This requires a scoring measure which is able to identify related regions across multiple sequences by comparing alignment columns instead of single amino acid pairs.

One of the most commonly used scoring measures for the construction of MSAs is the sum-of-pairs score (SP) [Thompson et al. 1999b]. This measure can be either used for scoring the residue similarity within an MSA (columns or regions), or between two columns of the same or different MSAs. The latter is especially important when two sequence alignments – so-called alignment profiles – have to be aligned in order to form a full MSA out of both alignments (Section 2.4.1).

The SP score $SP(i)$ of a column i simply corresponds to the sum of the log-odds scores S_{α_i,α_j} provided by a substitution matrix of all induced residue pairs. Pairs involving gap symbols are typically ignored in the calculations. Figure 2.7 illustrates the SP score calculation for an example column i obtained from an MSA of three sequences. This column contains three

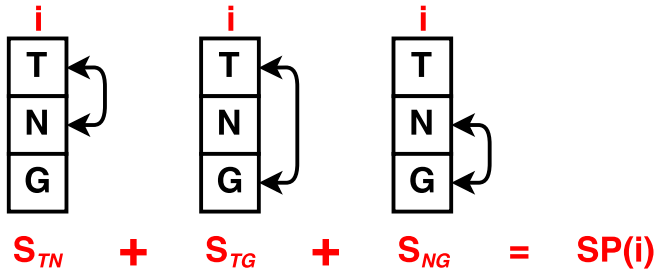


Figure 2.7: Example illustrating the log-odds scores summands of the SP score for a particular MSA column i. The shown example MSA is build of three sequences and thus induces $\frac{n\cdot(n-1)}{2} = \frac{3\cdot(3-1)}{2} = 3$ pairwise alignments, i.e., three different amino acid pairs are considered in the calculation of the SP score.

residues and thus contributes $\frac{n\cdot(n-1)}{2} = \frac{3\cdot(3-1)}{2} = 3$ residue pairs to its SP score. In particular, the SP score for the shown column i corresponds to $SP(i) = 1 \cdot S_{TN} + 1 \cdot S_{TG} + 1 \cdot S_{NG}$. By summing the column SP scores $SP(i)$ over all N columns of an MSA ($SP = \sum_i^N SP(i)$), one can obtain an MSA SP score which can be used, e.g., to compare and judge two alternative alignments (Chapter 5).

The SP score $SP(i,j)$ for two different profile columns i and j can be computed in a similar way. Instead of accumulating the score of all residue pairs within a single column, the SP measure builds the sum over the scores of all possible pairs that can be formed between each residue x from column i and each residue y from column j. For example, one can form six pairs of amino acids between the profile columns i (ACD) and j (AE): AA, AE, CA, CE, DA, and DE. The SP score for this profile columns then corresponds to $SP(i,j) = S_{AA} + S_{AE} + S_{CA} + S_{CE} + S_{DA} + S_{DE}$. This effectively measures the similarity between two profile columns based on their inherent residue types.

DP method for computing optimal MSAs

As explained in Section 2.2, the optimal global alignment of two sequences of length p and q can be computed with dynamic programming using a $p \times q$ matrix. This concept can also be applied to align an arbitrary number of sequences n of lengths l. However, this requires the calculation of a fixed number of operations per entry in an n-dimensional matrix with l^n entries [Carrillo and Lipman 1988]. In other words, one can think of the optimal global MSA of n sequences as a path inside an n-dimensional hypercube. Its sides correspond to the induced pairwise alignment of two sequences in the MSA. Finding the optimal solution thus has a computational complexity of $O(l^n)$. This makes the DP algorithm impractical for MSAs including more than a few sequences as illustrated in the following example. For an average sequence length of 250 amino acids, computing an MSA of three sequences already requires the calculation of $250^3 = 15,625,000$ entries. While computing this number of entries may still be feasible for modern computers, real world MSAs are usually build from much more sequences in order to provide meaningful information. A small MSA of only 10 sequences, e.g., already necessitates the computation of $250^{10} \sim 9.54 \times 10^{23}$ entries which cannot be performed by modern computers in a reasonable time.

To speed up the computation, Carrillo and Lipman [1988] proposed a strategy to reduce the number of calculations while still guaranteeing to find the optimal alignment. The central idea behind this strategy is that every MSA imposes pairwise sequence alignments between all of its n sequences, i.e., the sides of the n-dimensional hypercube as outlined above. These PSAs are projections of the MSA, i.e., the path followed inside the hypercube to obtain the MSA, into the two-dimensional space spanned between two sequences. Likewise, these PSAs also form bounds for the location of the optimal global MSA. This effectively restricts the number of "relevant" entries in the n-dimensional matrix to find the optimal solution which significantly reduces the number of computations.

The strategy by Carrillo and Lipman [1988] was later incorporated in the program `MSA` [Lipman et al. 1989] which employs dynamic programming in combination with the SP measure to compute multiple sequence alignments. Even though this strategy, as well as further modifications [Gupta et al. 1995], reduces the runtime complexity of `MSA` significantly, the remaining number of computation steps is still too large for computing MSAs with a reasonable number of sequences in an acceptable time. In fact, several publications have shown that computing the optimal MSA of n sequences with the DP algorithm under the SP scoring model is NP-complete [Wang and Jiang 1994, Just 2001, Elias 2006]. Most MSA programs used today thus rely on some kind of heuristic to speed up the computation. This comes, however, at the cost of uncertain MSA quality which can be problematic as outlined in the introduction of this thesis (Section 1.3).

Progressive alignment methods

Progressive alignment methods [Hogeweg and Hesper 1984, Feng and Doolittle 1987] are the most commonly employed heuristic-based approaches for constructing MSAs [Chatzou et al. 2016]. These methods first construct an initial alignment of the two most similar sequences and then progressively add the remaining sequences or sequence groups to the initial alignment by following the branching order of an evolutionary "guide" tree [Mount 2004]. At each inner node, a global pairwise sequence alignment is computed for either a pair of sequences, a pair of intermediate alignments (alignment profiles), or a sequence and an alignment profile. These pairwise alignments are typically computed using variations of the Needleman-Wunsch algorithm in combination with a particular scoring model such as the SP score which is able to measure the similarity between columns of different alignment profiles.

Figure 2.8 illustrates this construction principle for a small example alignment of four sequences. According to the evolutionary guide tree shown on the left hand side, the two most similar sequences are A and B. Thus, the first step of the progressive alignment process is to compute a PSA of A and B. The next node in the branching order of the guide tree starting from the currently computed PSA(A,B) is the guide tree's root node which aligns two groups of sequences (AB and CD). Processing this node is equal to the computation of a pairwise alignment of two alignment profiles, namely PSA(A,B) and PSA(C,D). In order to satisfy the requirements for processing the root node, the next step is to compute the PSA(C,D). The final MSA is then obtained by aligning the two profiles PSA(A,B) and PSA(C,D). As outlined in Section 2.4.1, this can be achieved by using the chosen alignment algorithm in combination with the SP score and by measuring the similarity between a column i from the profile PSA(A,B) and a column j from the PSA(C,D).

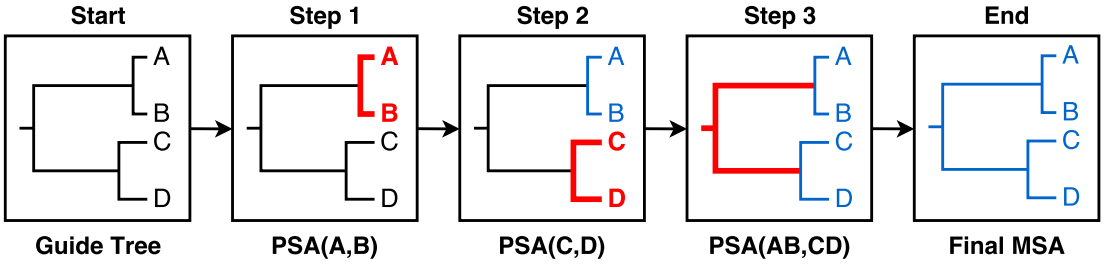


Figure 2.8: Flowchart illustrating the progressive sequence alignment process for a set of four sequences. The guide tree reflecting the similarity of the sequences is shown on the left hand side. The subtree being aligned in the current step is highlighted in red. Subtrees indicating finished intermediate alignments are colored in blue. By following the branching order of the guide tree, A is aligned with B in the first step followed by C and D in the second step. In the third step, the alignment profiles AB and CD are aligned to form the final MSA.

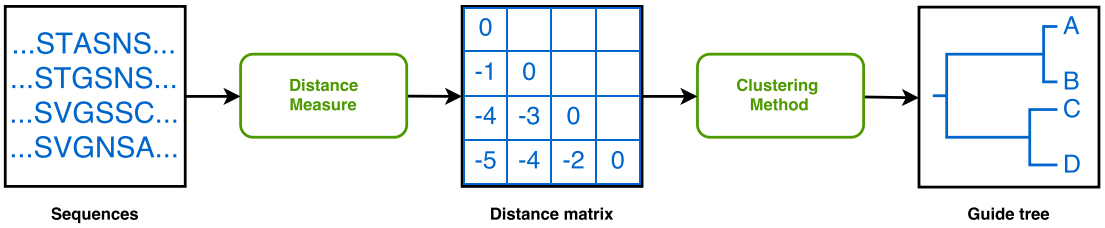


Figure 2.9: Flowchart illustrating the construction process of a guide tree for multiple sequence alignments. Given a sequence dataset and a particular distance measure, e.g., pairwise global similarity, a distance matrix is calculated. This matrix is then clustered with a specific clustering method like UPGMA to construct the final binary guide tree.

The *guide trees* used in progressive alignments are usually constructed by hierarchically clustering the sequence set using a distance matrix derived from pairwise sequence similarity scores (Figure 2.9). First, the similarities between all sequence pairs are calculated, e.g., using global PSA algorithms such as the previously described Needleman-Wunsch algorithm or by counting the number of identical subsequences of a given length k between two sequences (k-mers) [Edgar 2004b]. Subsequently, these scores are transformed into a distance matrix. The final guide tree is then computed from this distance matrix through hierarchical clustering, e.g., by employing standard methods for constructing phylogenetic trees such as the Neighbor-Joining [Saitou and Nei 1987] or UPGMA [Sneath and Sokal 1973] method.

The main advantage of progressive alignment methods is their speed. In contrast to the standard DP algorithm, even large MSAs can be computed in a reasonable time. However, this speed advantage comes at the cost of an uncertain alignment accuracy. Since the computed intermediate alignments (inner nodes in the guide tree) never change in later steps, mistakes made in earlier stages are propagated through the entire alignment process [Notredame et al. 2000, Wheeler and Kececioglu 2007, Sievers et al. 2011]. The accuracy of the final MSA thus strongly depends on the quality of the guide tree [Nelesen et al. 2008, Zhan et al. 2015]. In

order to mitigate these problems, many progressive alignment programs employ additional strategies such as iterative refinement steps or consistency-based alignment constraints. In the following, we describe these methods in more detail.

Iterative methods

Iterative alignment methods cope with the aforementioned problem that intermediate alignments remain fixed through the entire progressive alignment process resulting in potentially biased MSAs. Instead of computing a single final MSA only, these methods iteratively refine the final MSA by splitting the sequence set of the MSA in two groups which are subsequently re-aligned. During this process, the alignment structure within an individual group typically remains unchanged. If the re-aligned MSA has a higher score than the original one, it is returned as the result instead of the original MSA. This process is repeated multiple times until the MSA score does no longer improve or a certain number of iterations have been performed.

According to Wheeler and Kececioglu [2007], iterative MSA programs often select groups for re-alignment either randomly [Do et al. 2005] or based on the guide tree [Edgar 2004b, Katoh et al. 2002]. In the latter case, edges of the guide tree are repeatedly cut based on a particular strategy in order to divide the sequence set in two separate groups. `MUSCLE` [Edgar 2004b] for instance, iterates over all edges of the guide tree, while `MAFFT` [Katoh et al. 2002] visits edges in random order. Additional strategies were proposed, e.g., by Wheeler and Kececioglu [2007] including the so-called *exhaustive 2-cut*, the *random 3-cut*, and an on-the-fly method. The first strategy assigns a score to each edge representing its potential for an improvement through re-alignment to determine the visiting order of the edges. The random 3-cut divides the sequence set into three groups A, B, and, C and keeps the highest scoring realignment out of the three different merge orders (ABC, ACB, and BCA). With the on-the-fly method, also currently computed intermediate alignments are re-aligned by cutting edges to a grandchild or child before continuing the alignment process.

Iterative alignments methods often produce MSAs of higher accuracy when compared to standard progressive alignments without refinement steps. However, the gain in accuracy often depends on the number of iterations performed and thus typically comes at the cost of longer computation time. For the sake of higher accuracy, most commonly employed progressive aligners still apply at least a few iterative refinements steps. We will describe a representative selection of these methods in more detail in Section 2.4.2.

Consistency-based alignment methods

The accuracy of an MSA generated with the progressive alignment heuristic strongly depends on the quality of the guide tree. Due to the greedy nature of this approach, mistakes made in earlier stages when merging alignments directly affect the remaining alignment process and cannot be corrected later [Notredame et al. 2000, Wheeler and Kececioglu 2007, Sievers et al. 2011]. In other words, the processing order of the sequences in a progressive alignment may lead to a local optimum which may prevent the construction of a globally optimal MSA, i.e., an MSA that induces optimal pairwise alignments across all sequences. For example, starting the alignment process from a PSA of very distantly related sequences may prevent

very similar sequences from being optimally aligned to each other in the final MSA. Likewise, aligning very similar sequences first and then adding distantly related ones may completely break the optimal PSAs between these distantly related sequences.

According to Chatzou et al. [2016], the most common approach to avoid this effect and, in turn, to improve MSA accuracy is the usage of *consistency* [Notredame et al. 2000]. The key concept of this methodology is to employ optimal pairwise sequence alignments as constraints for the construction of a global MSA. By re-estimating the match scores of residue pairs based on information obtained from these PSAs, an adapted scoring model is derived. This scoring model can steer the alignment process towards generating MSA regions that are more "consistent" to the optimal PSAs of the involved sequences.

Kemena and Notredame [2009] report that consistency-based alignment programs generate MSAs with higher accuracy than iterative progressive methods on structural MSA benchmark datasets. This improvement in accuracy comes, however, at the cost of significantly higher computation time and memory consumption. Most implementations have a runtime complexity of $O(n^3)$ and a space complexity of $O(n^2)$ [Chatzou et al. 2016]. According to Kemena and Notredame [2009], aligning N sequences using consistency-based methods require N times more CPU time on average than standard progressive alignment programs.

The consistency methodology is incorporated into several MSA programs (e.g., `T-Coffee` [Notredame et al. 2000], `Mafft` [Katoh et al. 2002], `MSAProbs` [Liu et al. 2010], and `ProbCons` [Do et al. 2005]). Since `T-Coffee` is considered to be an archetype for consistency-based aligners [Chatzou et al. 2016], we describe this program in more detail in Section 2.4.2. An extensive review about other consistency-based methods can be found in [Kemena and Notredame 2009].

Phylogeny-aware alignment methods

Progressive alignment methods construct MSAs by successively performing pairwise alignments of two sequences or sequence groups until all sequences in the data set are added to the MSA. Each time a pairwise alignment of two groups is performed, additional gaps may be introduced into the final alignment that are fully penalized. Since pairwise alignments do not distinguish between insertion and deletion events, this can be problematic when introducing additional gaps into columns that already contain at least one gap. In this case, the additional penalty would account for the new deletions but existing and already penalized insertions would receive additional penalties as well.

Figure 2.10 illustrates this problem for a pairwise alignment of two example alignment profiles. Starting on the left hand side, two pairs of sequences are aligned by introducing gaps (red boxes) forming two different alignment profiles. Since pairwise alignments cannot distinguish between insertion and deletion events, the applied gap penalty accounts for both types of events. In the final MSA, additional gaps (red boxes) are introduced to form the pairwise alignment of the profiles computed in the previous step. Again, the penalties for these new gaps represent joint penalties for deletion and insertion events. Since the "theoretical" insertions (blue boxes) were already penalized during the construction of the alignment profiles, applying a full gap penalty in the profile alignment would "incorrectly" penalize the insertions for a second time. According to Löytynoja and Goldman [2005], this may result in over-aligned MSAs which effectively underestimate the true number of

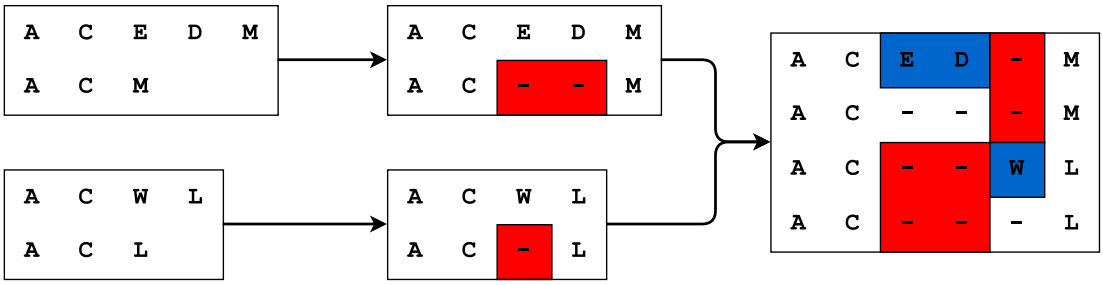


Figure 2.10: Flowchart illustrating the problem of penalizing insertions multiple times in a progressive alignment. On the left hand side, two pairs of sequences are shown that are aligned to form two profiles (middle). The gaps introduced in this step (red boxes) are fully penalized and account for insertion and deletion events. On the right hand side, the final MSA is shown which aligns the two profiles by introducing additional gaps (red boxes). Again, these gaps are fully penalized and thus include potential insertions as well as deletion events. Since the potential insertions (blue boxes) were already penalized during the construction of the profiles, this results in insertions being "incorrectly" penalized multiple times.

insertions. Thus, phylogeneticists often tend to manually refine similarity-based MSAs to generate MSAs that more likely reflect homology from an evolutionary perspective [Chatzou et al. 2016].

As reported by Löytynoja and Goldman [2005], traditional progressive alignment programs treat gaps as deletions and use heuristics to correct for multiply penalized insertions, e.g., by lowering the penalties for gaps that are introduced at sites already containing gaps as implemented in `CLustalW` [Thompson et al. 1994]. Even though this may prevent the alignment of non-homologous residues at insertion sites, the phylogenetic information contained in the indel events is discarded. This may result in substantial bias affecting phylogenetic analysis based on these MSAs [Löytynoja and Goldman 2005].

According to Chatzou et al. [2016], phylogeny-aware alignment methods address this issue by measuring the accuracy of an MSA based on the quality of its underlying phylogenetic model. For example, one of the first phylogeny-aware alignment methods `PRANK` [Löytynoja and Goldman 2005], directly avoids repeated penalties for insertions in order to produce MSAs that reflect the true phylogenetic tree more properly. Another example is the iterative phylogeny-aware alignment program `Saté` [Liu et al. 2009]. This program estimates the MSA supporting the highest-scoring maximum likelihood tree [Chatzou et al. 2016].

2.4.2 MSA programs

Over the last decades, a vast amount of MSA programs and algorithms have been proposed which compute protein MSAs out of a set of protein sequences. For instance, Chatzou et al. [2016] mentioned that over 100 alternative MSA methods have been developed over the last three decades.

These MSA programs mostly differ in their usage of different heuristics to approximate the optimal MSA which in turn affects their accuracy, supported number of sequences and runtime complexity. In this section, we thus focus on a representative selection of widely used

Program	Iterative	Consistency-based	Secondary Structure	Large-scale
ClustalW2	x	-	-	-
Clustal Ω	x	-	-	x
T-Coffee	-	x	-	-
MUSCLE	x	-	-	-
Decipher	x	-	x	-

Table 2.7: Widely used programs for the computation of protein multiple sequence alignments and their underlying methodologies. This includes whether a particular program uses some kind of iterative refinement, follows a consistency-based approach, makes use of secondary structure information, or can be employed for large-scale alignments containing thousands of sequences.

algorithms and programs for the construction of protein MSAs. Additionally, we included the program DECIPHER which represents one of the most recent MSA programs to date. An overview of these programs and their employed construction methodologies is given in Table 2.7.

ClustalW and ClustalW2

ClustalW [Thompson et al. 1994] is one of the oldest but also one of the most popular programs for progressively computing MSAs. In order to produce accurate alignments, ClustalW extends the standard progressive alignment procedure by a number of modifications. In particular, this includes an individual weighting scheme for scoring residue pairs based on relative sequence similarities, the usage of different substitution matrices for individual alignment stages, and an adaptive position-specific gap penalty model.

The guide tree used for the progressive alignment is constructed on the basis of pairwise sequence similarity scores using the Neighbor-Joining method [Saitou and Nei 1987]. ClustalW offers two methods for computing the pairwise similarity scores: The first method counts the number of identical k-tuples in the best PSA generated by a fast approximate method and subtracts a fixed penalty for each gap. The second method is reported to be more accurate. It counts the number of identical residues in an optimal global PSA. This PSA is constructed using a full dynamic programming approach in combination with a specific substitution matrix and an affine gap penalty model.

Analogous to the second method, the sequences or sequence groups defined by the guide tree are aligned using a full DP algorithm during the progressive alignment stages. In each stage, the positions A_i and B_j between two sequences or sequence groups A and B are rated as the average of the weighted substitution matrix scores of each residue in A_i paired with each residue in B_j. Pairings with gaps receive a score of zero. In order to associate gaps with the worst possible score, the matrices are rescored to contain only positive values. The aforementioned weights of each sequence depend on its distance in the guide tree to the root node. This distance is also used as a similarity measure between sequences or sequence groups which determines the scoring matrix used for each alignment stage.

The penalties for introducing gaps are adapted according to a set of rules. Each gap is scaled by the average score for a substitution event and the similarity of the involved sequences in percent. The length of the sequences and their differences also affect the gap penalties. Additionally, `ClustalW` uses residue-specific and position-specific gap costs. For instance, the gap penalties are locally reduced in hydrophilic stretches to encourage gaps in loop regions. Likewise, introducing gaps in regions already containing gaps is less penalized.

In 2007, Larkin et al. presented a new `ClustalW` version called `ClustalW2` [Larkin et al. 2007]. `ClustalW2` provides a set of extensions to the original `ClustalW` program in order to improve the program's runtime and the accuracy of the generated MSAs. In particular, this includes a faster guide tree computation based on the UPGMA method [Sneath and Sokal 1973] and the option to iteratively refine the generated MSA. The latter option can be applied to intermediate alignments or the final MSA. It successively removes and realigns each sequence in the MSA and subsequently checks whether the re-alignment results in an improved MSA score.

T-Coffee

The greedy nature of progressive alignment strategies can introduce a severe bias since alignment errors made in early stages are propagated to the final MSA. In order to mitigate this problem, `T-Coffee` [Notredame et al. 2000] considers all sequences during each alignment stage, instead of only those involved in the current intermediate alignment. In other words, `T-Coffee` aims at constructing MSAs of which the induced pairwise alignments match their optimal counterparts.

First, `T-Coffee` creates a library of pairwise sequence alignments. For each pair of sequences, a global PSA and the ten best non-overlapping local PSAs are computed. These computations use the previously described `ClustalW` software and the program `Lalign` [Huang and Miller 1991] of the FASTA package [Pearson and Lipman 1988], a variant of the Smith-Waterman algorithm (Section 2.2.1). The pairwise residue matches represented by these alignments are then used as constraints for the further alignment process. In order to designate their importance for the final alignment, each residue pair also receives a weight equal to the percentage of preserved residue pairs in the corresponding PSA.

In a second step, the information provided by the global and local PSAs is merged by stacking duplicates into a single library by summing up their weights. The resulting primary library is then extended by analyzing the transitive pairwise alignment relationships of the residues. For each residue match between two sequences A and B, `T-Coffee` checks the corresponding residue matches in the PSAs of A with sequence C, and B with sequence C. If these matches are identical, i.e., refer to the same residuum in C, the weight for the particular residue match between A and B is increased. This process is repeated until all transitive relationships between all sequence pairs AB to all other sequence $C \notin AB$ are analyzed.

On basis of this extended library, `T-Coffee` performs a progressive alignment according to the method used in `ClustalW`. First, a guide tree is computed using the Neighbor-Joining method based on a distance matrix. It is generated from sequence similarity scores obtained through pairwise alignments. By following the branching order and using the aforementioned DP algorithm, sequences or profiles are aligned using the position-specific weights stored in the extended library. When aligning two profiles at inner nodes of the guide tree, each column is rated by the average of the position-specific weights referring to the column's position.

Since the position-specific scores already reflect the gap penalties used in the construction of the underlying PSAs, T-Coffee does not require a further gap penalty model. Hence, all gap penalties are set to zero when performing the progressive alignment.

T-Coffee was reported to produce more accurate alignments than other methods such as Dialign2 [Morgenstern 1999] or ClustalW [Thompson et al. 1994]. However, this accuracy comes at the cost of a high computation time: The runtime complexity of T-Coffee is $O(N^2L^2)+O(N^3L)+O(N^3)+O(NL^2)$ with N denoting the number of sequences in the MSA and L the average sequence length. The individual terms correspond to the computation of the primary library, the extension of the library, the construction of the Neighbor-Joining guide tree, and finally the computation of the progressive alignment.

MUSCLE

MUSCLE [Edgar 2004a, Edgar 2004b] is another popular progressive alignment tool employing iterative refinement for improved MSA accuracy and a novel objective function, the so-called log-expectation (LE) score. This scoring function is a modified version of the log-average scoring function [Ohsen and Zimmer 2001]. It takes the amino acid composition as well as the frequency of gaps within the columns into account. MUSCLE computes an MSA in three stages: a draft progressive stage, an improved progressive stage, and a final refinement stage. At the end of each stage, an MSA is available. MSAs created in later stages are, however, usually more accurate.

In the draft progressive stage, a first MSA is computed emphasizing speed over accuracy. First, a distance matrix representing the pairwise similarities of the sequences is computed using a fast but potentially inaccurate k-mer distance measure. Afterwards, the distance matrix is used to construct a guide tree by employing the UPGMA method [Sneath and Sokal 1973]. Analogous to other progressive aligners, the MSA is then computed by following the branching order of this guide tree. At each internal node, an intermediate alignment of the corresponding two alignment profiles is generated by comparing profile columns using the aforementioned log-expectation score. Through this, MUSCLE effectively favors alignment columns with few gaps over those containing a large number of gaps.

At completion of the draft progressive stage, a first MSA is available. The quality of this MSA is, however, questionable due to the usage of a probably suboptimal guide tree induced by the approximative nature of the k-mer distance measure. In the second stage, the so-called improved progressive stage, MUSCLE thus re-estimates the guide tree based on the MSA of the draft progressive stage. First, MUSCLE computes the pairwise residue identity within each pairwise alignment induced in this MSA. The computed values are then converted into a distance matrix by applying a Kimura correction [Kimura 1983] which accounts for multiple substitutions at a single site. A new guide tree is then derived from this distance matrix, again using the UPGMA clustering method, and an additional progressive alignment is performed as described above. However, only subtrees with different branching order to the initial guide tree are progressively re-aligned in order to save computation time.

In the third and last stage, MUSCLE performs a user-specified number of refinement steps to further improve the MSA. At each step, an edge, which has been visited in decreasing distance from the root, is deleted to split the guide tree into two separate subtrees. The resulting alignment profiles are then re-aligned. If the new alignment provides a higher SP score than the previous MSA, the new alignment is kept or otherwise discarded.

According to Edgar [2004], MUSCLE quickly produces MSAs with a comparable or superior accuracy compared to other alignment methods (e.g., ClustalW [Thompson et al. 1994], T-Coffee [Notredame et al. 2000], and MAFFT [Katoh et al. 2002]) on state-of-the-art alignment benchmarks such as BAliBASE or SABmark. Even without refinement, the quality of the MSAs generated with MUSCLE was reported to be indistinguishable from those computed with T-Coffee or MAFFT. Edgar also declared that this version was the fastest of all tested methods. It only required 7 minutes for the computation of an MSA of 5,000 sequences with an average length of 350 residues.

Clustal Omega

Clustal Omega [Sievers et al. 2011] is the latest version of the aforementioned Clustal programs which aims to align several hundreds of thousands of sequences quickly. In order to handle this huge number of sequences, Clustal Omega employs a fast approximate guide tree construction followed by a progressive alignment based on pairwise alignments of profile hidden Markov models.

According to Sievers et al. [2011], the direct computation of all pairwise similarity scores between N sequences in order to obtain a distance matrix to be used as an input for guide tree construction typically has a runtime and space complexity of $O(N^2)$. To reduce the high costs that comes with the alignment of huge sequence datasets, Clustal Omega uses the mBed [Blackshields et al. 2010] method in combination with k-means or UPGMA [Sneath and Sokal 1973] clustering to approximate a guide tree in $O(N \log N)$. Instead of computing all pairwise similarities, mBed only computes the similarity between a number of *seed* sequences t sampled from the data set and all non-seed sequences. The similarity scores can quickly be computed using the k-mer measure which is also implemented in ClustalW2. Afterwards, each sequence is replaced by a vector of length t with coordinates representing the similarity scores obtained for the sequence vs. seed comparisons. Clustering the resulting vectors with k-means or UPGMA then yields the final guide tree.

The multiple sequence alignment is computed progressively by following the branching order of the computed guide tree. At each intermediate alignment stage, Clustal Omega aligns two profile hidden Markov models (HMM) [Eddy 1998] using the HHalign package [Söding 2004]. For increased accuracy, Clustal Omega can also iteratively refine its guide tree and HMMs.

According to Sievers et al. [2011], Clustal Omega produces alignments of a comparable quality to other methods for smaller numbers of sequences. For larger sequence datasets, it was reported to outperform other MSA programs in terms of accuracy and computation time.

DECIPHER

DECIPHER [Wright 2015] is one of the most recent programs for computing MSAs. Similar to the aforementioned tools, DECIPHER computes MSAs using the progressive alignment strategy and iterative refinement. Unlike other popular alignment programs, however, DECIPHER also incorporates structural information in the alignment process. The program performs four stages to produce an MSA.

In the first stage, secondary structure prediction is performed on each sequence using the GOR algorithm [Garnier et al. 1996] (Version IV). This results in each residue being annotated with its probability to belong to a specific secondary structure conformation within its sequence, namely helix (H), β-sheet (E) or coil (C).

In the second stage, a rough guide tree is computed based on shared k-mer distances similar to MAFFT and MUSCLE. In contrast to these programs, DECIPHER only considers k-mers that occur in the same order in both sequences. According to the author, this effectively mitigates the problem that particular k-mers are expected to occur frequently by chance in longer sequences but comes at the cost of higher computation time. The resulting distance matrix is clustered by single-linkage clustering to obtain an initial guide tree.

By following its branching order, two profiles are globally aligned using a variation of the Needleman-Wunsch algorithm in combination with a modified version of MUSCLE's profile sum-of-pairs (PSP) objective function. This objective function additionally encourages conserved regions of matching secondary structure using the previously predicted secondary structure information. Gaps are penalized in DECIPHER by a position-specific gap opening penalty and an extension penalty proportional to the length of a gap raised to a power. The opening penalty varies depending on the surrounding residues and their likelihood to occur near gaps. In addition, gap opening and extension costs are adapted linearly depending on the similarity of the sequence profiles to restrict gaps between closely related sequences.

During the third stage, a new UPGMA guide tree is calculated based on the Hamming distance of each sequence pair in the alignment obtained from the second stage. For each node difference between this UPGMA tree and the previous guide tree, the underlying sequence profiles are re-aligned. This stage is repeated for a fixed number of iterations.

In the last stage, the obtained MSA is further refined. The UPGMA tree is successively split at all edges that separate the sequences into two groups with a distance of at least 70%. These groups are then re-aligned.

2.5 Analysis and comparison of multiple sequence alignments

Another important application in the context of sequence alignments is the assessment of the accuracy of alignment programs and the resulting alignments. While PSA algorithms guarantee to produce the optimal alignment given a particular scoring model (Section 2.2), the usage of heuristics in MSA algorithms typically results in unknown MSA accuracy and thus requires verification (Section 2.4). Since the optimal MSA of an arbitrary sequence set under a given scoring model is unknown, the accuracy of MSA programs is normally assessed using standardized MSA benchmarks.

MSA benchmark datasets (e.g., BAliBASE 3.0 [Thompson et al. 2005], OXBench [Raghava et al. 2003], SABmark 1.65 [Van Walle et al. 2005]) provide specific sets of sequences and corresponding reference alignments. These reference alignments are typically manually curated or verified and are thus considered to be correctly aligned. Often, these reference alignments are designed to represent specific alignment tasks. This includes, e.g., the alignment of structurally related sequences (BAliBASE 3.0) or very dissimilar sequences (SABmark 1.65).

For each benchmark set of sequences, sequence alignments can be computed using arbitrary MSA programs and corresponding parameters. The obtained *alternative alignments* can then be compared with the *reference alignment* by employing a particular comparison measure in order to assess the accuracy of the generated alignment and in turn the accuracy of the alignment program and/or its parameters.

2.5.1 MSA benchmark datasets

Commonly used MSA benchmark datasets are, for instance, BAliBASE 3.0 [Thompson et al. 2005], OXBench [Raghava et al. 2003] and SABmark 1.65 [Van Walle et al. 2005]. These benchmarks are also available in a single convenient benchmark collection called `bench` [Edgar 2009a] which covers a broad range of different alignment scenarios.

BAliBASE 3.0 is one of the most widely used MSA benchmarks and provides 386 MSAs categorized in five different sets. Each set represents a specific MSA use case, e.g., a set of very divergent sequences (Reference 1) or sequence families that are aligned to a distantly related sequence (Reference 2). The MSAs in each set were generated using a combination of sequence- and structure-based methods with manual refinement [Edgar 2010]. SABmark 1.65 provides two sets of MSAs, a "Twilight Zone" set (209 MSAs) and a "Superfamilies" set (425 MSAs) which are derived from a consensus of SOFI and CE [Boutonnet et al. 1995]. While the sequences in the first set share a maximum similarity of 25%, the second set contains sequences with a maximum similarity of 50%. The underlying sequences were selected using fold information from the SCOP database and thus possess known structure. OXBench provides a set of 395 structural MSAs in total constructed using `STAMP` [Russell and Barton 1992] and 3D structural information from the 3Dee database [Siddiqui et al. 2001].

2.5.2 MSA comparison measures

Several measures for the comparison of two alternative sequence alignments have been proposed over the last years. In this thesis, we also present additional MSA comparison measures later in Chapter 5. One of the simplest methods is to compute the sum-of-pairs (SP) score for both alternative alignments and to consider the alignment with the higher score as the "better" alignment. This method can also be accompanied with a certain gap penalty model. However, the SP method is not able to quantify the differences between both alignments in a meaningful way since the SP score for the optimal alignment is unknown and thus no normalization can be applied.

Other measures represent the differences between a reference and a target alignment more accurately. For example, the *BAliBASE total column score* [Thompson et al. 1999a] expresses the difference between two alignments by the fraction of the number of identically aligned columns in both alignments and the total number of columns in the reference alignment. Some measures assess the alignment differences on a per residue level. The *q-score* measure [Edgar 2004b], e.g., counts the number of identically aligned residue pairs in both alignments and relates this count to the total number of aligned pairs in the reference alignment. This measure is also known as the *BAliBASE SP score* [Thompson et al. 1999a] or the *Developer score* [Sauder et al. 2000]. Likewise, the *Modeler score* [Sauder et al. 2000] corresponds to the fraction of the total number of identically aligned residue pairs in both alignments and the total number of aligned residue pairs in the target alignment.

The aforementioned column- or residue-based measures are widely used especially when evaluating the accuracy of MSA programs on the basis of state-of-the-art MSA benchmark datasets. However, these methods do not report the magnitude of the per-column or per-residue differences of the alignments.

Shift score measure

Another well-established alignment comparison measure is the so-called *shift score* by Cline et al. [2002]. Unlike the aforementioned per-residue comparison measures, the shift score not only counts the number of differently aligned residues but also takes the magnitude of their alignment differences into account: Given two alternative pairwise alignments of sequences A and B, the shift score analyzes the *shift* $\delta(x)$ of each residue at index x in sequence A by comparing the indices (y_1 and y_2) of its aligned residues in sequence B in the two alternative alignments. Likewise, the shift values of the residues of sequence B are measured with regard to sequence A.

Figure 2.11 illustrates this principle based on two alternative example PSAs. Shown on the left hand side is the reference PSA with two aligned residues being highlighted (orange). The index of the leucine residue (L) in sequence A is denoted with x and the index of the aligned methionine (M) with y_1. On the right hand side, an alternative PSA is shown. In contrast to the reference alignment, the same leucine residue of sequence A is aligned with an isoleucine in sequence B at index y_2. The shift value for this leucine residues thus corresponds to $\delta(x) = y_1 - y_2 = 6 - 9 = -3$. The final *shift score* $\Delta(x)$ is then computed based on the shift values $\delta(x)$ according to the following equation:

$$\Delta(x) = \frac{1+\epsilon}{1+|\delta(x)|} - \epsilon \tag{2.10}$$

The parameter ϵ allows to adjust the scoring range of the shift score measure between $-\epsilon$ and 1. In fact, ϵ defines at which shift value the *shift score* drops into the negative range. The usage of $\epsilon = 0.2$ as suggested by the authors results, e.g., in residues with a total shift of 5 receiving a score of $\Delta(x) = 0$. Residues with larger shifts than 5 would receive negative shift scores successively approaching $-\epsilon$. Perfect matches are always rated with the highest shift score of $\Delta(x) = 1$ regardless of the chosen ϵ-value. Notably, the shift score is not defined ($\Delta(x) = 0$) for alignment constellations where a residue is aligned to a gap symbol.

Figure 2.11: Example of a reference PSA (left) and an alternative PSA (right) illustrating the *shift score* measure by Cline et al. [2002]. In the reference PSA, the leucine (L) at index x in sequence A is aligned with a methionine at index y_1 in sequence B. The same leucine is aligned in the alternative PSA with an isoleucine (I) at index y_2. The *shift* value $\delta(x)$ of the leucine between the two alignments thus corresponds to $\delta(x) = y_1 - y_2 = 6 - 9 = -3$.

Chapter 3
Serious games for bioinformatics

3.1 Introduction

Playing games is a central activity in human life across all societies, genders, and age classes. Games are structured forms of playing behavior defined by specific rules and goals. Especially *digital games* that use computing hardware such as a personal computer or a video game console became more and more popular over the last years. In 2015, 42% of all citizens in Germany older than 14 years played computer and video games [Bitkom Research 2015]. Another study performed by the entertainment software association (ESA) showed that in 2017 at least one person in 65% of the U.S. households regularly plays video games [ESA 2017]. According to this study, the average gamer is 35 years old.

People play games primarily for entertainment but some games are also designed with the intention to achieve additional goals while still eliciting fun to the players. The prominent board game Monopoly, e.g., was originally intended to serve as an educational tool to demonstrate the negative effects of private monopolies [Orbanes 2007]. In light of the large and steadily increasing number of computer gamers, digital games thus offer a great potential for motivating and involving humans even in non-gaming tasks.

Games that are designed with this potential in mind are often loosely denoted as *serious games* or mistakenly associated with the term *gamification*. In the context of this thesis, we use the definitions by Dörner et al. to clarify these terms. They define a serious game as "a digital game created with the intention to entertain and to achieve at least one additional goal (e.g., learning or health)" [Dörner et al. 2016, p. 3]. These additional goals are denoted as *characterizing goals*. In contrast, gamification describes the process of transferring "game methodologies or elements to non-game applications and processes" [Dörner et al. 2016, p. 3]. The outcomes of gamification thus do not necessarily represent an actual game [Dörner et al. 2016, p. 4].

Serious games that are explicitly designed to engage players in specific non-game tasks are called *games with a purpose* (GWAP) [Dörner et al. 2016, p. 6]. They combine the intrinsic motivation of gaming with the concepts of *crowdsourcing* and *citizen science*. In citizen science, research tasks are outsourced to a volunteer crowd of amateur or non-professional "scientists" [Hand 2010]. Cooper et al. [2010] introduced a specific subcategory of games with a purpose called *scientific discovery games*. According to Cooper et al. [2010], these types of games "translate computationally difficult scientific problems into puzzles" and provide "game-like mechanism for non-expert players to help solve these problems".

Citizen science approaches and especially scientific discovery games can be powerful tools to address computationally expensive problems in biology. The scientific discovery game Phylo [Kawrykow et al. 2012], e.g., demonstrates that the time-consuming process of manual MSA refinement can successfully be outsourced even to non-experts by transforming the

task into a puzzle-style game. However, their approach does not support the refinement of protein MSAs. The authors also did not evaluate if their game is fun to play and delivers a true *game experience*.

According to Dörner et al. [2016, p. 11], game experience is the subjective experience of true gaming. Providing the player with a true game experience while still achieving the characterizing goals is essential for the success of a serious game. The concept of game experience includes several dimensions, e.g., fun, flow, immersion, challenge, tension, and positive as well as negative emotions. One of the most important dimensions is game flow. It represents a player's feeling of being absorbed by the game. As of Dörner et al. [2016, p. 11], game flow is characterized by an exclusive concentration on the game and the feeling of control over the game. Other aspects include being immersed by the game, facing clear goals, and receiving immediate and consistent feedback. To enable the player to enter the state of flow, a serious game thus must maintain an appropriate balance between the task difficulty and the player's skill level.

The players' subjective game experience can be measured, e.g., using the game experience questionnaire (GEQ) [IJsselsteijn et al. 2008, Nacke 2009]. This questionnaire assesses game experience using 36 items referring to seven categories: immersion, flow, competence, tension, challenge, positive and negative affect. The items are based on a Likert scale with five possible answers ranging from "not at all" to "extremely". The authors also offer a shorter version called in-game experience questionnaire (iGEQ). This version is designed to assess a player's game experience during a game session. Thus, it only contains two items per GEQ category.

Inspired by Phylo, we developed our own scientific discovery game called Bionigma for improving the quality of protein MSAs. Unlike Phylo, Bionigma is especially designed to deliver a true game experience to motivate and attract a large number of players. For this purpose, we followed an iterative development process. In each iteration, we implemented a new game version and evaluated the resulting prototype in various user studies. Based on the feedback and insights obtained from the different evaluations, we successively improved our game concept. As shown by the results of our follow-up studies, we could successfully improve its capabilities to elicit fun and to deliver a true game experience with each iteration.

In this chapter, we first present successful examples for citizen science approaches and games with a purpose addressing problems in biology and bioinformatics. The remainder of this chapter describes the different developmental stages of our scientific discovery game approach for solving and improving protein MSAs in detail. These parts also include detailed evaluations and discussions of each developmental stage.

3.2 Related work

As mentioned above, citizen science describes the process of solving scientific problems by a volunteer crowd of amateur or non-professional scientists [Hand 2010]. The success of this concept often depends on the number of participating volunteers and thus motivation is a crucial aspect. A popular method to motivate and encourage people for these tasks are computer games, i.e., games with a purposes. In this section, we present a selection of successful citizen science (game) approaches in detail. This includes two of the most

popular citizen science approaches to date (SETI@home and Galaxy Zoo) that demonstrate the overall usefulness of this concept. In particular, we focus on scientific discovery games that address important problems in biology.

3.2.1 Crowdsourcing and citizen science

SETI@home and Folding@home

One of the most popular citizen science projects is *SETI@home* [Korpela et al. 2001]. This project aims at analyzing radio signals using home computers to search for signs of extraterrestrial intelligence. While performing other tasks for different scientific projects, the Arecibo radio telescope passively collects a vast amount of radio data. These datasets could contain hints about extraterrestrial intelligence but their analysis requires an immense computational effort. To address this problem, SETI@home divides the data into smaller chunks. These chunks are then analyzed using distributed computing by employing private computers from volunteers. Unlike other citizen science projects, the volunteer supporters thus only provide SETI@home with computational resources but do not manually interact with the data.

The success of this approach led to follow-up projects such as *Folding@home* [Shirts and Pande 2000] which extends this citizen science concept to the molecular biology domain. Folding@home uses the idle processing resources of computers provided by volunteers to discover the process of protein folding. The obtained information is especially useful for medical research and drug design.

Galaxy Zoo

Galaxy Zoo [Lintott et al. 2008, Willett et al. 2013, Willett et al. 2017, Simmons et al. 2017] is a citizen science project that aims at the morphological classification of galaxies in millions of pictures by using the natural visual pattern recognition capability of humans.

The users determine the shape of galaxies presented to them and classify certain features. This is achieved by answering a questionnaire with a finite number of answers based on a decision tree. For each available answer, Galaxy Zoo provides the user with example images to support them in the decision process. In Galaxy Zoo 2, e.g., the root node of the decision tree refers to the question "whether the galaxy is either 'smooth', has 'features or a disk', or is a 'star or artifact'" [Willett et al. 2013]. Depending on the user's answers, the next steps ask for further details about the chosen galaxy type or exit the classification when the leaf nodes in the decision tree are reached. In order to collect reliable classifications and to remove outliers, the user results are weighted based on different criteria such as the community consensus [Simmons et al. 2017].

Since the initial start of Galaxy Zoo in July 2007 and across different project phases, over one million morphologies for galaxies have been collected [Simmons et al. 2017]. Galaxy Zoo thus demonstrates the huge potential of employing crowd-sourcing concepts for the task of visual pattern recognition.

Project Discovery

Project Discovery is a citizen science project embedded in the massively multiplayer online role-playing game Eve Online [2017]. It outsources different research tasks to the players in return for in-game rewards. The first research task in Project Discovery was the identification of the subcellular localization of proteins in cells based on microscope images [Project Discovery 2017]. According to Marx [2015], this information can be used, e.g., to obtain clues about the function of the proteins and to create cellular maps. The latter can also help when characterizing disease states.

Conforming to Marx [2015], subcellular protein locations are usually detected with spatial proteomics techniques such as Immunofluorescence. However, the resulting images typically have to be manually reviewed by experts in order to properly identify and categorize the subcellular locations of the proteins [The Human Protein Atlas Blog 2016]. While this may be feasible for a small number of images, the effort for processing huge image datasets such as the millions of images taken in the Human Protein Atlas (HPA) project is immense [Uhlén et al. 2005, Uhlen et al. 2010, Uhlén et al. 2015]. For this reason, the cumbersome classification task was outsourced to a volunteer crowd of Eve Online players of Eve Online [2017].

Eve Online is a massively multiplayer online role-playing game (MMORPG) with a persistent world set up in a space scenario. The players can fly around in their space ships and visit star systems while participating in in-game activities such as mining, piracy, manufacturing, trading, exploration, and combat. The aforementioned classification task is embedded as a mini-game within Eve Online. For increased immersion, it is also interweaved into Eve Online's story line. The players are recruited by the Sisters of EVE (SoE) to assist in the identification of biological samples obtained from Drifters, a mysterious faction in Eve Online. These samples are in fact the microscopy images obtained within the HPA's subcellular protein atlas project. Likewise, the task is the identification of patterns of protein distributions in human cells.

According to the information provided on the Project Discovery [2017] website, the mini-game shows the original microscope image to the players. Proteins in question within this image are highlighted in green. Blue regions indicate cell nuclei and red lines represent microtubules that fill the cytoplasm. The players may activate different filters or zoom the image to focus on specific details. The player's task is to classify the shown image according to different categories illustrated by example images. If none of the depicted categories match, the player can choose to declare the sample as abnormal indicating that the image cannot be properly categorized.

For each solved task, the players earn small amounts of Interstellar Kredits (ISK), experience points, and Analysis Kredits that can be spend in the main game. The magnitude of the rewards depend on the player's accuracy rating. This rating represents the reliability of the player to accurately classify the images and serves as some sort of quality criteria.

As described on the Project Discovery [2017] website, this accuracy is measured in two different ways. Before the players are allowed to classify real samples, they have to process a set of pre-classified examples. Based on this dataset, a first accuracy rating is calculated by comparing the pre-classifications with the player's solutions. As soon as a player is allowed to process real samples, the player's accuracy rating is adjusted for each accomplished task

after a consensus community solution is found. Depending on the deviation of the player's solution from this consensus, the player's accuracy rating is either increased or decreased. If such a consensus for the player's current task exists, it is shown to the player after submitting his own solution.

Already two months after launch of Project Discovery in March 2016, 40,000 players helped in the analysis of 250,000 images of stained tissue samples resulting in over 4.5 million individual protein location annotations [Peplow 2016]. Project discovery thus provides further evidence for the usefulness of citizen science approaches and employing humans for otherwise computationally expensive tasks.

3.2.2 Scientific discovery games in biology

Foldit

Foldit [Cooper et al. 2010a] is a multiplayer scientific discovery game that aims at finding accurate solutions for the protein folding problem. According to [Lesk 2010, p. 123], many approaches to predict a protein's 3D structure try to reproduce the interatomic interactions in the protein. Through this, an energy model can be derived that allows to compute the free energy of any conformation. The 3D structure of a protein can then be predicted by finding the minimum of this conformational energy function.

As reported by Cooper et al. [2010], predicting the 3D structure of proteins is still a largely unsolved problem at least for larger proteins. The authors relate this to the enormous number of degrees of freedom that have to be taken into account when searching the extremely large free energy landscape of the protein. However, the authors of Foldit hypothesized that the spatial reasoning of humans could help in sampling the conformational space and determining suboptimal conformations.

Improperly folded protein conformations are depicted in Foldit using a simplified 3D visualization of the protein's structure. The backbone of the protein is shown as a single object hiding its underlying atoms. In contrast, the structure of the amino acid side chains including the position of the atoms are fully visible. Depending on whether a side chain is hydrophilic or hydrophobic, the side chain mesh is either colored in blue or orange. The color of the various backbone regions indicate their energy. Low energy regions are colored in green and those with high residue energy that should be improved are colored in red. Additionally, Foldit visually highlights certain regions of interest. This includes, e.g., energetically frustrated areas that could be improved, hydrogen bonds, or interatomic repulsion and cavities.

In order to improve the energy of the shown protein structure, the players can directly manipulate the 3D structure. This can be achieved by pulling specific parts of the protein in the desired direction. Additionally, Foldit provides the players with different automatic tools based on algorithms from the Rosetta structure prediction methodology [Rohl et al. 2004]. To constrain the structure of the protein during these "automatic moves", the players can add "rubber bands" or "freeze" specific degrees of freedom. The current score of the played level (the negative of the Rosetta energy) is shown at the top of the game screen along with the player's current ranking in comparison to other players. The latter provides further motivation for the players to improve their score.

To engage players with no previous knowledge in molecular biology, Foldit provides a tutorial to learn the aforementioned game mechanics. In addition, Foldit features the creation of interaction macros [Cooper et al. 2011]. These so-called recipes are collected in a "cookbook" and are available to all Foldit players.

Recent studies showed that some solutions found by Foldit players even outperformed their computer-based counterparts [Khatib et al. 2011a, Khatib et al. 2011b]. This demonstrates that Foldit's approach for the optimization of protein structure prediction results actually works.

Phylo

Phylo is a scientific discovery game for improving DNA multiple sequence alignments [Kawrykow et al. 2012]. Like all MSA problems, the computation of DNA multiple sequence alignments is also NP-complete. Thus, heuristics are traditionally used to speed up the computation process. Likewise, the resulting alignments may be suboptimal which can negatively affect further applications based on these alignments. Since humans typically outperform computers in detecting misalignments, an important step to improve the quality of sequence alignments is manual refinement. However, this is a cumbersome task that often requires the knowledge of experts. To address this problem, Phylo abstracts the alignment problem in the form of a puzzle game to enable even non-experts to successfully refine DNA alignments.

Similar to visual MSA editors (Section 5.2.2), Phylo represents a DNA sequence alignment in a grid-like manner. Each row in this grid refers to a specific DNA sequence. The nucleotides inside these sequences are depicted as colored squares with the color indicating their type. Empty cells represent gaps in the alignment. The players can improve the alignments by aligning identically colored squares to increase the number of matches while simultaneously reducing the number of mismatches and gaps. This is achieved by moving single squares or a rectangular selection of squares to the left or right through a mouse drag gesture. Neighboring squares that collide with the current selection during movement are also moved in the current movement direction.

Instead of providing the players with all sequences from the beginning, the players align the full set of sequences in successive steps according to the branching order of an evolutionary tree. This is similar to the progressive alignment method presented in Section 2.4.1. At each alignment stage, the players must achieve a given minimum alignment score called par in order to proceed with the next stage or – in the last stage – to finish the level. The par value of the current stage as well as the player's current score and personal highscore is shown at the top of the screen as numbers.

The score of the current alignment stage is computed using an identity-based substitution model and affine gap penalties. The total score is based on the similarity of a particular sequence and its ancestor defined by the evolutionary tree, i.e., the consensus sequence of the inner node. For each pair of matching squares between the sequence and the ancestor, the player receives a reward of +1 point. Likewise each pair of mismatching squares is penalized by −1. Additionally, Phylo penalizes each individual gap and each gap extension by a certain amount. Terminal gaps are not penalized.

Depending on the finally achieved alignment score, the players increase their rank in a global ranking list. This list contains four different categories: the number of times a player matched the par value, beat the par, holds the highscore, and the total sum of these values. This mechanism provides another motivation factor for the players to achieve the highest score per level and to solve additional puzzles. Additionally, the player may choose the puzzles from a set of eight disease categories. Each category represents alignments that are associated with a particular disease such as cancer or blood and immune system diseases. This is another strong motivation factor for the players. It suggests that playing Phylo actually helps to cure dangerous diseases.

According to the statistics shown at the Phylo website[1], a total number of over one million MSAs had been solved to date (24.10.2017). This indicates that Phylo was able to successfully attract players to help in refining sequence alignments. According to the results presented in the initial publication of Phylo [Kawrykow et al. 2012], the player-refined MSAs outperformed their computer-generated counterparts in most cases. This demonstrates that the player-refined alignments are actually useful.

Nonetheless, this promising approach has some limitations. According to [Kawrykow et al. 2012], Phylo can only deal with DNA sequences and only supports the refinement of small alignment subsets of 8 sequences with a few number of nucleotides. The size of the playing field is fixed which may constrain the players in their movements. Additionally, the number of supporting information or interactions is very limited. There is also no further evidence besides the actual number of improved MSAs that Phylo is actually fun to play and delivers a true game experience.

3.3 First prototype

The goal of our first game prototype [Hess et al. 2014b] was to provide an initial puzzle game version for protein MSA refinement with comparable features to Phylo [Kawrykow et al. 2012] and some additional improvements such as increased alignment size. Additionally, we wanted to investigate whether this kind of puzzle-style citizen science game is able to deliver a true game experience and enables non-experts to refine protein MSAs.

3.3.1 Approach

In line with Phylo and other visual MSA editors (Section 5.2.2), our prototype visualizes a sequence alignment in a grid-like manner similar to an abacus (Figure 3.1). Each row in this grid refers to a particular protein sequence in the alignment. Its corresponding amino acids are represented by differently-colored circular discs, while gaps in the alignment are depicted as empty space between the acids. For simplicity, we did not model similarities between different amino acid types (e.g., different substitution scores) in this first prototype. Instead, we employed a simple identity-based scoring model like that used in Phylo and color-coded each individual acid type by a unique color (Figure 3.2).

For each pair of aligned amino acids of identical type (identical colors), the player receives a fixed reward, while pairs of mismatching amino acids are penalized with a fixed negative score. Gaps are penalized row-wise by an affine gap penalty model using fixed gap opening and extension penalties (Section 2.1.2). The heights of the match reward and the penalties

[1] http://phylo.cs.mcgill.ca/, last accessed 24.10.2017

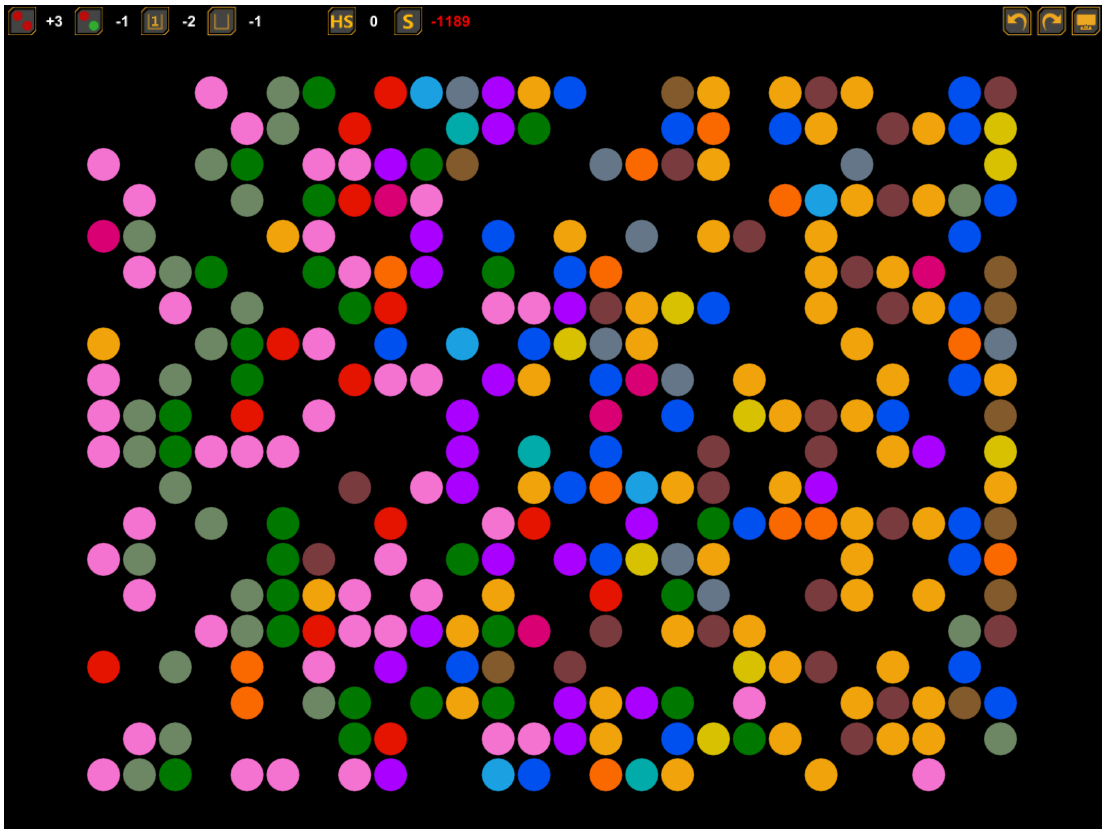


Figure 3.1: The game interface of our first game prototype. The alignment grid is shown at the center. Each row corresponds to a specific protein sequence. The amino acids inside the proteins are depicted as circular disks. The different colors indicate the different acid types. The icons and values shown at the upper left corner depict scoring information about the current level (From the left to the right: match reward, mismatch penalty, gap opening and extension penalty, level highscore, and current alignment score). The upper right corner displays the undo/redo and menu buttons.

Figure 3.2: The color scheme used in the first prototype to represent the 20 different standard amino acids with unique colors.

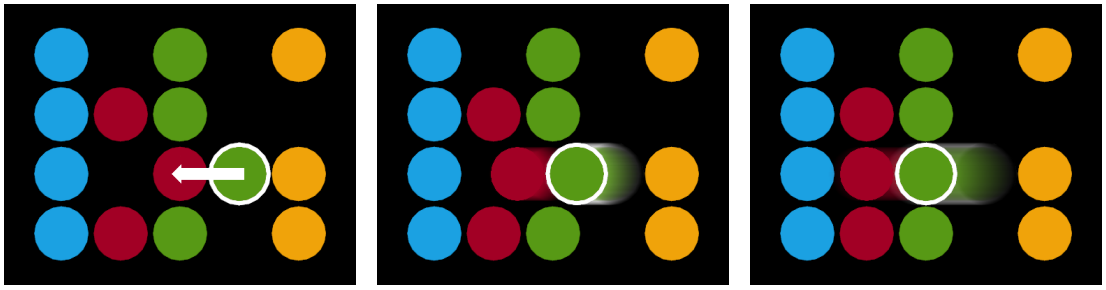


Figure 3.3: Aligning by drag and drop. The green disk with the white outline is dragged one position to the left. It collides with the neighboring red disk which is in turn also moved one position to the left.

are adapted for each individual game level. The settings for the currently played level are shown at the upper left corner of the game user interface. In the level shown in Figure 3.1, e.g., a match is rewarded with $+3$ points, while a mismatch receives a penalty of -1. The introduction of a gap results in a penalty of -2 and extending an existing gap by one unit costs 1 point.

Like existing MSA programs (Section 2.4.2), our prototype computes the total alignment score S based on the sum-of-pairs scoring measure (Section 2.4.1). In particular, the rewards and penalties for all pairs of aligned amino acids in the alignment and the penalties received for each individual gap are accumulated. The score of the current alignment is also shown at the top of the user interface (S) next to the level-depending highscore threshold (HS). This threshold represents the minimum alignment score a player must achieve to complete a level.

To solve a given level, the player must improve the alignment by aligning identically colored residues while simultaneously reducing the number of mismatches and gaps. A single amino acid can be aligned through a mouse drag gesture. This moves the selected acid highlighted by a white ring either to the left or right side (Figure 3.3). Neighboring amino acids that collide with the acids being currently moved are simply “pushed” in the movement direction. During the movement phase, the score of the current alignment stage is recalculated in real-time. This enables the player to determine if the movement results in a score improvement or not.

Beside this movement interaction, the player can revert and restore each performed alignment change by clicking on the undo and redo buttons shown at the upper right corner of the user interface. Additionally, the shown alignment can be zoomed in order to get an overview of a large alignment or to focus on specific alignment regions.

To support easy learning of the game mechanics and controls, we also implemented a short stepwise tutorial. In each step, the player learns a new aspect of the game mechanics and can directly try out the learned content inside the tutorial level. This is contrary to Phylo which only provides the player with a written tutorial in combination with screenshots.

3.3.2 Evaluation

Methods

We evaluated our early game prototype in a user study with a convenience sample of 20 persons (6 women and 14 men; age: 22 to 34 years). The participants were colleagues and students from the biology ($n = 11$), computer science ($n = 8$), and educational science domain ($n = 1$). Notably, five participants had no prior experience in computer games.

The purpose of this evaluation was to test the prototype concerning the quality of the tutorial, the game mechanics, and the game experience. We implemented 10 different puzzles with increasing difficulty based on artificial sequence alignments. With each difficulty step, we increased the number of sequences and their lengths, reduced the similarity of the sequences, and adapted the scoring scheme. For practical reasons, the largest levels were limited to 20 sequences with a length of 16 amino acids.

Each test person had a maximum of one hour of playing time. Beside this time limit, there were no further limitations. Thus, the participants were allowed to stop playing whenever they wanted as well as to skip the tutorial or certain levels. The participants were encouraged to report their immediate game impressions and to identify issues while playing the game. After playing, the participants had to answer a questionnaire consisting of 38 items based on a Likert scale with five possible answers, ranging from “not at all” to “extremely”.

The first 14 items represented a variation of the in-game experience questionnaire (iGEQ) [IJsselsteijn et al. 2008, Nacke 2009] (Table A.1). As outlined in the introduction, this questionnaire measures game experience by seven categories: immersion, flow, competence, tension, challenge, positive and negative affect. For compatibility reasons to our game (e.g., there is no storyline), we substituted the original iGEQ items one, five, 13, and 14 with the GEQ items 14, 28, 13, and 22 of the same category.

The items 15 to 29 were related to usability aspects regarding the design of the user interface and the controls according to the ISO 9241/(1)10 norm. These items were designed in correspondence to the Isometrics Questionnaire [Gediga et al. 1999]. This part of the questionnaire is shown in Table A.2. The last 9 items addressed the usability and helpfulness of the tutorial (Table A.3).

We analyzed the different items by coding their corresponding answers with integer values between 0 (“not at all”) and 4 (“extremely”). The obtained results are shown in the following sections as box plots. Bottom and top of the shown boxes correspond to the lower (Q_1) and upper quartile (Q_3) of the answer distribution, while the horizontal line inside the box represents the median $\tilde{x}$. The mean μ is depicted as a small dot. The whiskers show the minimum and the maximum of the underlying distribution.

Results - Game experience

The results of our user study for the seven GEQ categories measuring the game experience delivered by our prototype are shown in Figure 3.4. Despite its early stadium, our first prototype received overall positive ratings with only slight variations in the answer distributions. Our prototype performed very well in the category of positive affect with a median of $\tilde{x} = 3$ indicating that the players had fun playing the game and felt pleased. They also had

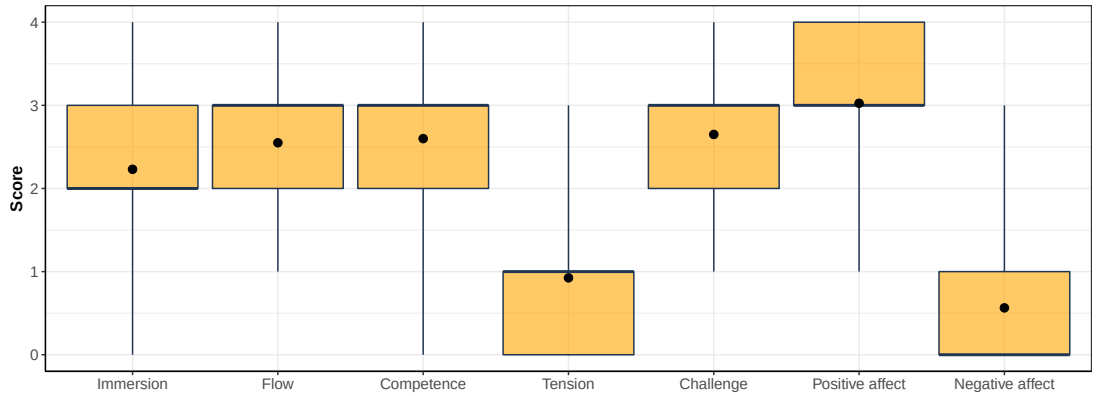


Figure 3.4: Box plot showing the distributions of answers received for each individual GEQ category.

a substantial feeling of competence and challenge in the game also indicated by medians of $\tilde{x} = 3$ and mean values of $\mu \sim 2.6$. The majority of the participants reported a loss of their sense of time during the play which relates to a considerable flow ($\tilde{x} = 3$, $\mu = 2.55$).

In the immersion category, our prototype performed moderately with a median of $\tilde{x} = 2$ and a mean value of $\mu = 2.23$. While the majority of the participants were aesthetically pleased by the prototype ($\mu = 2.7$), they were only moderately impressed ($\mu = 1.74$). With regard to negative impressions, our prototype showed also very good results. The majority of the players were neither bored nor found the game tiresome. This results in a mean of $\mu = 0.57$ and a median of $\tilde{x} = 0$ in the category of negative affect. Additionally, they experienced only a very low level of tension ($\tilde{x} = 1$, $\mu = 0.61$).

Results - Tutorial

The evaluation of our prototype's tutorial showed also reasonable results (Figure 3.5). All but one participant played the tutorial in order to familiarize themselves with the game mechanics. The majority of the participants found the tutorial comprehensive ($\tilde{x} = 3$, $\mu = 3.3$). Likewise, most participants rated the tutorial as well structured ($\mu = 3.06$) and helpful ($\mu = 2.79$). They also had the feeling that they learned all relevant aspects of the game mechanics ($\mu = 3.06$). However, the answers referring to these three categories vary stronger ($Q_1 = 2$, $Q_3 = 4$) in contrast to the comprehensibility rating of the tutorial.

The possibility to instantly practice the learned content in the tutorial level was also considered to be very positive ($\tilde{x} = 4$, $\mu = 3.5$). Some participants rated the tutorial length as too long ($\tilde{x} = 1$, $\mu = 0.97$) and would appreciate a shorter version with fewer text components.

Results - Usability and game mechanics

The results of the usability and game mechanics parts of our user study are shown in Figure 3.6. The design of the graphical user interface (GUI) was rated very well with a mean of $\mu = 2.95$. The participants were very pleased with the screen layout and they rated the graphical user interface to be very comprehensive. The ratings of the controls showed even better results with a median of $\tilde{x} = 4$ and a mean of $\mu = 3.37$. The participants stated that the controls were very intuitive, easy to learn, good to memorize, and precise.

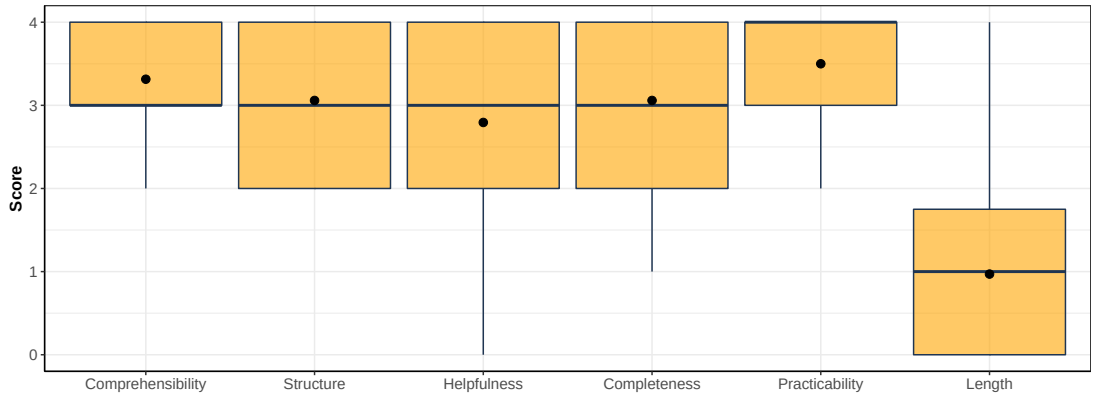


Figure 3.5: Box plot showing the distributions of answers received for the individual aspects of the tutorial.

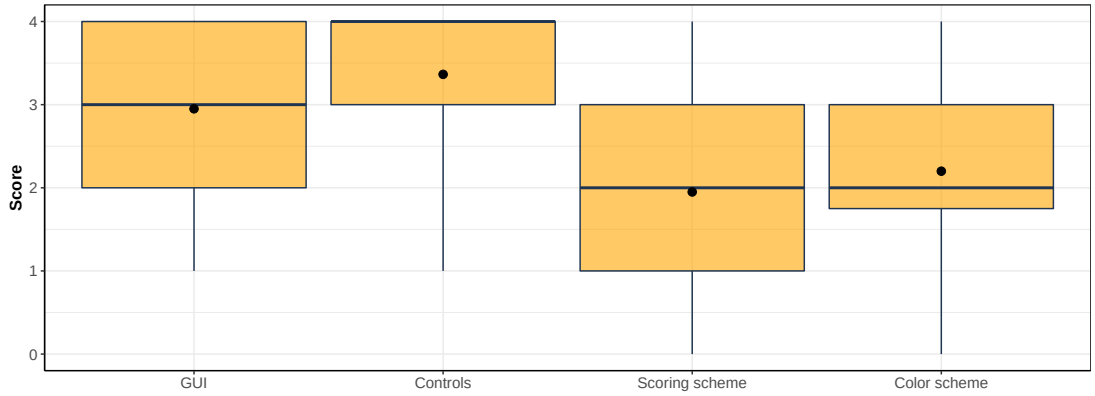


Figure 3.6: Box plot depicting the answer distributions for the assessed usability and game mechanics aspects.

Contrary to the tutorial, the comprehensiveness of the scoring system received only moderate ratings ($\mu = 1.95$). Some of the participants had problems to identify the reasons for an increase or decrease of the score, especially when gaps were involved. Another problem reported by some test users was the lack of discriminability of some acid colors like different types of green. As a result, the rating for the color scheme was only moderate with a mean of $\mu = 2.2$.

Discussion

The results of our user study demonstrate that our puzzle game prototype already performed reasonably well regarding usability and game experience. However, there exists no “ground truth” to which our results can be compared since the game experience of comparable games such as Phylo was not assessed by the authors. Furthermore, comparing our game to a purely entertaining game or to professional sequence alignment editors would be equally unfair. Still, the results of this first study can serve as a reference for follow-up studies.

Some major issues revealed in our study are the comprehensibility of the scoring scheme and the discriminability of the different acid colors, especially in alignments with twenty different amino acid types. According to flow theory [Csikszentmihalyi 2000], a negative effect of these issues on game experience would have been reflected in low flow and high tension scores. As illustrated in Figure 3.4, this is obviously not the case. In addition, some participants felt constrained in their game control due to the lack of a multi-selection feature. The players' immersion is another aspect that should be improved in later game versions.

3.3.3 Conclusion

Our first puzzle game prototype represented an improved game approach for the manual refinement of protein sequence alignments using an integration of the concepts of citizen science and gaming. We implemented this prototype with a major focus on usability, visually appealing appearance, and fun of gaming. The evaluation of our prototype regarding these aspects showed promising results. The participants rated the game prototype as visually appealing, controllable, and challenging. In spite of the early developmental state of this game version, the players already had much fun in playing it, especially in advanced levels. Still, the simple identity-based scoring model is a major limitation since it does not properly reflect the state-of-the-art for rating protein alignments.

3.4 Second prototype

Based on the user feedback and insights obtained during the development of our first prototype (Section 3.3.2), we improved our game concept in various ways. From a player's perspective, major issues of the first game version were the reported low player immersion and the lack of more sophisticated interactions. The simple identity-based scoring model was a major issue from a biologist's point of view since it did not reflect the state-of-the-art for protein MSAs. In order to address these limitations, our second prototype provides a fully redesigned game interface, new interactions, and implements a real-world scoring model for protein MSAs.

3.4.1 Approach

The redesigned game interface of our second prototype is shown in Figure 3.7. The alignment grid is still depicted in the center of the game window with each row representing a particular protein sequence. The corresponding amino acids are shown again as colored circular disks. Likewise, the undo/redo buttons and the menu are shown at the top right corner of the screen.

To increase the player's immersion and to provide a more visually appealing game experience, we embedded the game scene in a science-fiction-styled interface border. Additionally, we replaced the scoring information panel at the top of the interface with a more intuitive progress bar. This bar shows the current level score as numbers and by the filling level (blue) of the progress bar. The scoring threshold of the currently played level that must be fulfilled to complete the level is shown as a marker on the right side of the progress bar. During movement, the bar also depicts in real-time the potential reward (green) or penalty (red) the player earns for the current alignment state as well as the resulting total alignment score. This helps the player to directly determine the reward or penalty for the current move supporting him or her in the alignment process.

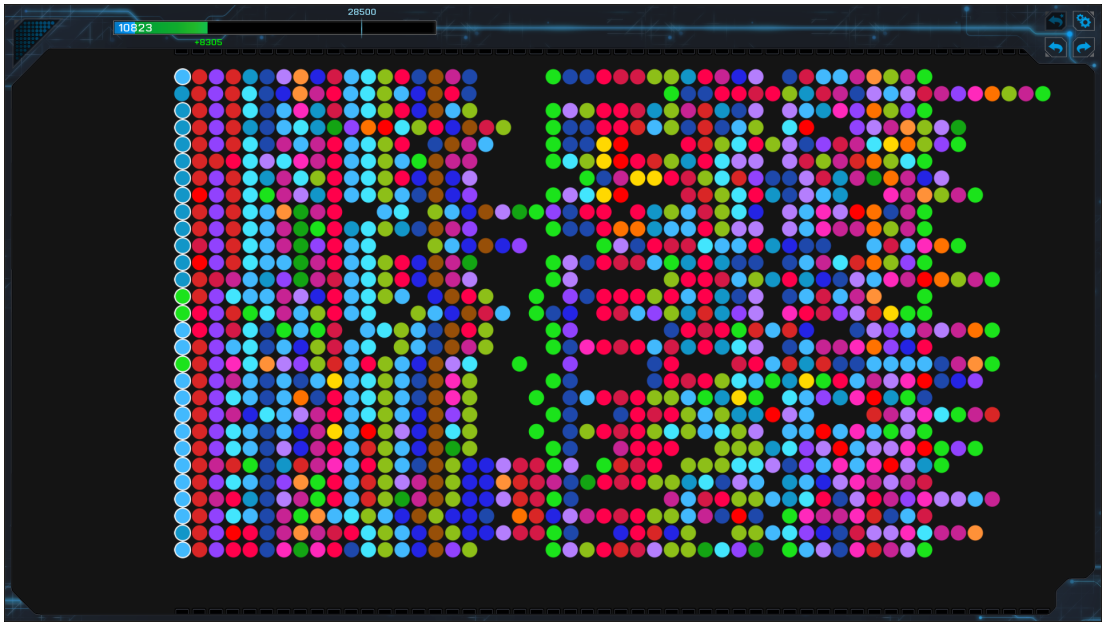


Figure 3.7: The game interface of our second prototype. The alignment grid is shown at the center. Each row corresponds to a specific protein sequence. The amino acids inside the proteins are depicted as colored circular disks indicating the different acid types. Similar colors like different tones of blue indicate similar amino acids that receive increased BLOSUM62 substitution scores when being aligned. The current alignment score is shown at the top in the form of a progress bar with the level scoring threshold being depicted as a vertical mark. The buttons shown at the top right corner provide access to the game menu and to the undo/redo functions.

Furthermore, we implemented an additional function to revert the current level to its high-scoring state shown together with the basic undo/redo function at the upper right corner of the interface. Through this function, the player can test different alignment constellations without loosing their best-scoring state or requiring the cumbersome process of manually saving and loading specific level states.

We also implemented new selection interactions to support the players in the alignment process. The player can select an arbitrary number of amino acids by either selecting individual amino acids per left click, drawing a selection rectangle via a drag gesture, or by double clicking on specific amino acids. The latter allows the user to select all amino acids of the same type within the current column. Additionally, we added selector buttons for each column at the top and bottom of the alignment window to enable the player to select entire columns. Similar to our first prototype, the selected acids can be moved in real-time to the left or right side using a mouse drag gesture. Likewise, colliding amino acids are pushed in the current movement direction.

Besides these enhancements, we redesigned the color scheme of the amino acids in order to cope with the real-world MSA scoring model used in this game version. Instead of an identity matrix, we selected the commonly used BLOSUM62 matrix [Henikoff and Henikoff 1992a] for representing substitution events between different amino acid types. Substitution

Figure 3.8: The color scheme used in our second prototype for depicting the 20 different standard amino acids. Similar colors, like different tones of red, indicate similar amino acids that provide scoring bonuses when being aligned.

matrices typically encode mutation probabilities between two amino acids in the form of rounded log-odds scores derived from observed substitutions in aligned sequence datasets (Section 2.1.1, Chapter 4).

According to these scores, some amino acid substitutions are considered to occur more likely than other substitutions. Substitutions between identical acid types occur most often in aligned sequence datasets and thus usually receive the largest scores in substitution matrices (e.g., $S_{WW} = 11$ in BLOSUM62). Additionally, substitutions between amino acid types with similar physico-chemical attributes are considered to be more likely than substitutions between largely different acid types [Lesk 2013, p. 184]. Substitutions between similar amino acid types thus also receive larger scores in most substitution matrices including BLOSUM62. For this reason, we manually designed our new color scheme to reflect these similarities by similar colors that are still distinguishable for detecting identical amino acid types.

The resulting color scheme is shown in Figure 3.8. For instance, substitutions between the aliphatic amino acids isoleucine (I), leucine (L), methionine (M), and valine (V) receive positive scores according to BLOSUM62. Thus, we colored these amino acids using different tones of red.

3.4.2 Evaluation

Methods

We evaluated our second game prototype in another user study with a sample size of 28 persons (9 women, 18 men, and 1 no indication; age: 7 to 61 years). All test persons were visitors at the public day of the GameDays 2014[2] and had no specific background in scientific fields such as biology or computer science. The focus of this study was to assess the capabilities of our game prototype to deliver a true game experience.

For this test setting, we implemented five different puzzles with increasing difficulty and size based on artificial sequence alignments. The smallest level represented an MSA of five sequences with a maximum length of nine residues. The two largest levels contained 29 and 38 sequences with maximum sequence lengths of 45 and 37 amino acids, respectively. The test persons were allowed to play the game without any restrictions. Additionally, they

[2] http://www.gamedays2014.de/, last accessed 24.10.2017.

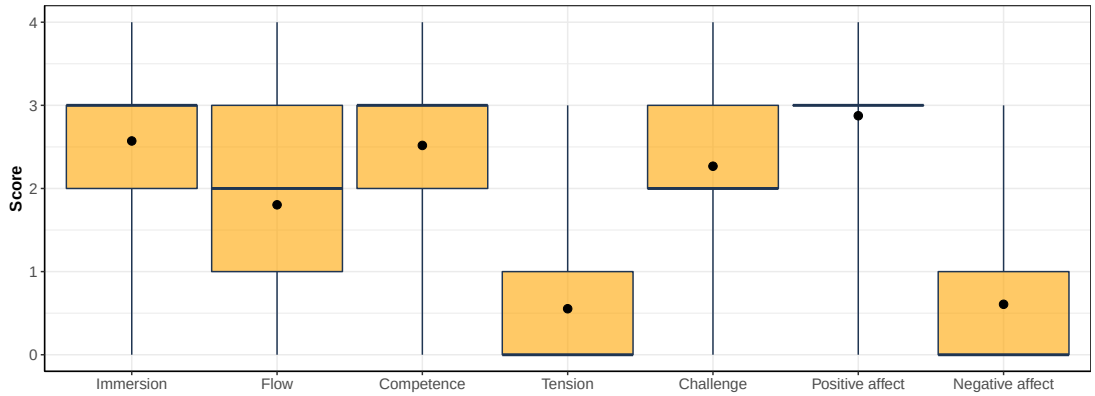


Figure 3.9: Box plot depicting the answer distributions of the different GEQ categories obtained in the user study of our second prototype.

were encouraged to report their immediate game impressions while playing the game. After playing, the participants had to answer our adapted version of the in-game experience questionnaire (iGEQ) [IJsselsteijn et al. 2008, Nacke 2009] (Table A.1).

Results

The results obtained for the different GEQ categories based on the iGEQ questionnaire are shown in Figure 3.9. Again, our game prototype received overall positive ratings. In the immersion category, our second prototype performed on average better than our previous prototype with median and mean values of $\tilde{x} = 3$ vs. $\tilde{x} = 2$ and $\mu = 2.57$ and $\mu = 2.23$, respectively. The main improvement in this category was achieved for the question if the players felt impressed. Here, the mean was $\mu = 2.32$ compared to $\mu = 1.74$ obtained in the evaluation of our first prototype. In general, our new interface design received a positive feedback and was reported to be visually appealing.

A slight drop could be observed in the challenge category. Unlike its predecessor, our second prototype was considered to be slightly less challenging with median and mean values of $\tilde{x} = 2$ vs. $\tilde{x} = 3$ and $\mu = 2.27$ vs. $\mu = 2.65$, respectively. This indicates that our second prototype delivered a more balanced game experience regarding the difficulty of the provided game levels. While our second game version achieved almost identical ratings to our first prototype in the categories competence ($\tilde{x} = 3$, $\mu = 2.52$) and negative affect ($\tilde{x} = 0$, $\mu = 0.61$), it performed substantially better in the tension category with a median and mean of only $\tilde{x} = 0$ and $\mu = 0.55$. In the category positive affect, our second game version also received positive ratings indicated by a median of $\tilde{x} = 3$ and a mean of $\mu = 2.88$. This demonstrates that the player had fun playing the game and felt pleased.

Notably, our second prototype received worse ratings compared to its predecessor in the flow category. Several participants reported no loss of their sense of time during the play indicated by a median of $\tilde{x} = 2$ instead of $\tilde{x} = 3$ and mean values of $\mu = 1.8$ versus $\mu = 2.55$.

Discussion

The results of our study showed that our second puzzle game prototype overall delivered a true game experience. In comparison to our first prototype, we could improve the ratings of the GEQ categories immersion and tension, and the different difficulty levels of the tested game levels were also considered to be more balanced. These improvements were primarily achieved by our new graphical user interface through its more game-like look and feel, the immediate information about scoring rewards and penalties, and the new selection interactions.

Notably, our second prototype achieved worse ratings than its predecessor in the flow category. However, this drop can be related to environmental circumstances as mentioned by some participants. While the participants of our first study played the game in a quiet environment, the test persons of our second study were stronger affected by external disturbances caused by the environmental setting at the public day of GameDays 2014.

On average, the employed BLOSUM62-based scoring model and the corresponding color-coded acid similarities were considered to be transparent and comprehensible. However, the participants also reported that some pairings of amino acid types provided rewards even though they were differently-colored (e.g., histidine (green) and tyrosine (orange)). Likewise, some participants were confused that similarly-colored amino acids yield negative scores instead of rewards. For instance, proline (P) and glycine (G) are both colored in green but substitutions between them are rated with a score of -2. This indicates that our color scheme still needs refinement in order to represent amino acid similarities more properly.

3.4.3 Conclusion

Our second game prototype implements several improvement over its predecessor. In particular, this includes a more visually appealing and game-like interface, more sophisticated interactions, and a real-world scoring model for protein MSAs. These improvements enabled our second prototype to receive higher iGEQ ratings on average than our first game version. In particular, our study demonstrated that this game version is able to deliver a true game experience to the player.

In addition, the usage of the BLOSUM62-based scoring model in combination with the color-coded acid similarities finally enabled us to implement game levels for the refinement of real protein MSAs, i.e., the initial purpose of our citizen science game approach. However, according to the participants' feedback received during our user study, the color scheme still needs further refinement to reflect the amino acid similarities encoded in BLOSUM62 more properly and in more intuitive way.

3.5 Bionigma

The final version of our scientific discovery game approach is called *Bionigma*[3], a portmanteau of the words **bio**logy and e**nigma**, the Greek word for "riddle". Bionigma is based on our second game prototype but implements several further improvements to overcome its

[3]Bionigma can be played without any charge. Versions for Windows, Linux, and Mac OS X are available as download at our game website http://www.bionigma.de, last accessed 24.10.2017.

Figure 3.10: The in-game visualization of a small protein sequence alignment in Bionigma. Each row of the shown grid refers to a specific sequence in the alignment. The colored game tokens represent the different amino acids.

limitations as well as additional features for increased game experience. In the following sections, we describe Bionigma in detail. The remainder of this section presents the results of our third major user study and discusses the accuracy of the player-refined protein MSAs.

3.5.1 Approach

Game concept

Bionigma represents a full scientific discovery game for enabling non-experts to support biologists in refining real protein sequence alignments while having fun and undergoing a true game experience. Similar to our previous game prototypes, the protein MSA problem is abstracted in the form of a puzzle game representing an MSA as a grid filled with colored game tokens. Each row in the grid represents a particular protein sequence and the colored tokens its corresponding amino acids (Figure 3.10).

By aligning similar game tokens, the player refines the real MSA and earns rewards increasing the total level score. Likewise, adding additional gaps or aligning dissimilar game tokens results in score penalties. In order to align tokens, the player can select single tokens through a left-click, identical tokens in the same column trough a double-click, entire columns using the column selector buttons shown at the top or bottom of the game interface, or multiple tokens by drawing a selection rectangle. The selected tokens can then be aligned by moving them to the left or right side using a mouse drag gesture. Similar to our game prototypes, neighboring tokens are also moved in the movement direction on collision with already moving tokens. In Bionigma, the score improvement or loss of a move is shown in real-time

at the scoring bar at the top of the screen and by a small numeric value shown below the mouse cursor. This allows the user to directly assess the benefit or loss of an alignment action.

To finish a level, i.e., to accomplish a Bionigma mission, the player must achieve a certain minimum score. The height of this threshold depends on whether the player played the level the first time or tries to improve the already achieved score. In the former case, the game uses a preset threshold that corresponds to the score of an existing MSA solution minus a certain offset defined by the level creator. This ensures that a certain level can be mastered by most players and prevents player frustration. In the latter case, the threshold represents the highscore achieved by the Bionigma community for the given level.

After reaching a level's minimum score, the player can decide to finish the current level or to further refine the level to beat the current highscore. In both cases, the player receives point rewards depending on the difficulty rating of the played level. These point rewards increase the player's rank in Bionigma's global player ladder as well as his ranking for the finished level. Beating the highscore of a level (mastering a level) rewards the player with a huge amount of bonus points. Simultaneously, the previous master of the level loses his or her bonus for holding the highscore resulting in a rank decrease of this player in the global ladder. Notably, players may decide to further improve already played MSAs at any time to (re-)claim the level mastership. This mechanic adds an additional challenge to the game that encourages players to further improve their MSAs to deliver even better solutions.

To provide the players with an easy way to learn the aforementioned game mechanics, we implemented a step-wise tutorial. In this tutorial, the player is guided through the different game stages by explanations and direct instructions given in text form. Additionally, all relevant areas on the screen are highlighted to focus the player on its current task. Similar to the tutorial of our first prototype, the player can directly try out the learned content in the game. The tutorial starts with the selection of a new level at the mission hub and guides the player with instructions and tips through the process of refining a small example alignment. In a second step, the player learns more advanced interactions techniques.

In order to collect the refined MSA data as well as provide the player with the aforementioned ranking information, Bionigma uses a client/server approach. While the Bionigma client represents the actual game containing the full game logic, the Bionigma server provides the clients with new game levels (new protein MSAs), receives and stores the refined MSA results from the players, and holds and provides ranking information of the players. To play Bionigma, the player thus must create a personal account. However, this only requires the selection of a unique nickname, a password, and entering the player's birth date. The latter information is used for statistics only.

Main menu

After login, the player enters Bionigma's main menu shown in Figure 3.11. By clicking on the buttons on the left hand side of this menu, the player can switch between different screens such as the *Mission Hub*, the player's *profile*, the *ladder*, or the *options menu*. The latter provides different options to configure the game including graphics and sound settings or changing the game language. Currently supported languages are German and English. The ticker at the top of the screen shows general information about the game such as the overall progress of the community in improving MSAs.

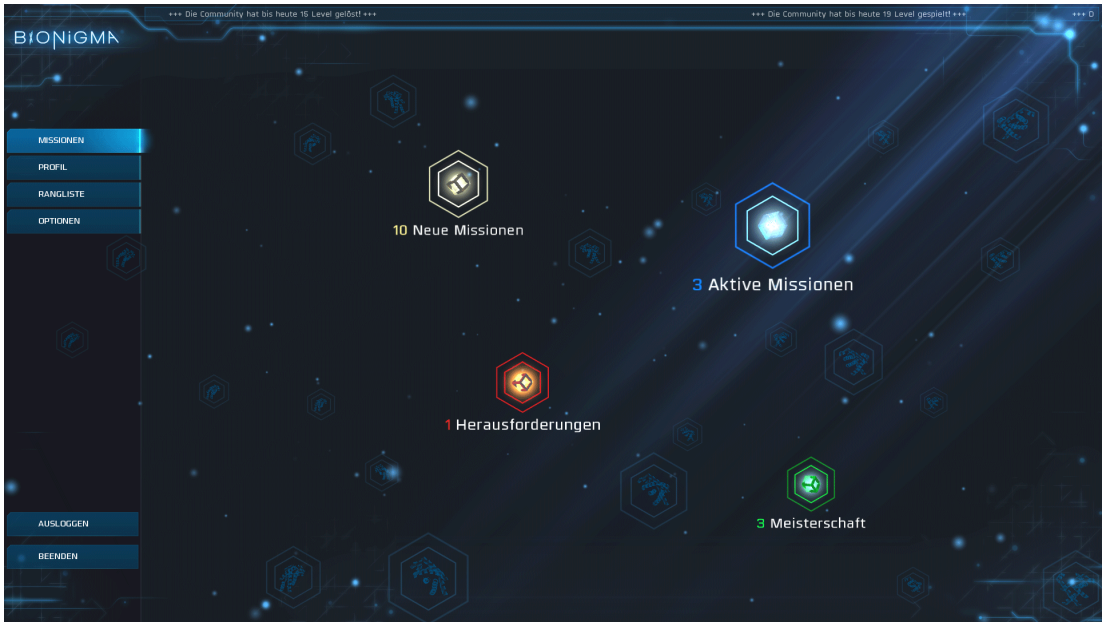


Figure 3.11: Bionigma's main menu currently showing the mission hub screen. Additional screens including the player's profile and the ladder can be accessed using the buttons on the left hand side. The buttons shown at the center of the mission hub provide access to new levels (white), list played but not yet finished levels (blue), and show the list of mastered (green) or challenge levels (red).

The profile screen shows different statistics about the player's overall game progress. This includes, e.g., the total number of played and finished levels as well as the number of currently mastered levels. Likewise, the ladder shows the player's rank in the global ranking list in comparison to other players of the Bionigma community.

The most important screen is the mission hub opened as standard screen after login (Figure 3.11). It enables the player to access a list with new levels of different difficulty (white button) and to get access to already played but still unfinished levels (blue button). In order to encourage the player in finishing levels, the current version of Bionigma limits the player to only store a total amount of 10 unfinished or new levels before additional levels are provided by the server. The green button opens the list of mastered levels, while the red buttons lists all challenge levels where the player does not hold the highscore and may try to (re-)claim the level mastership.

Visualization

After selecting a new level or an existing one for further refinement, the player enters the actual puzzle game (Figure 3.10). As outlined above, the MSA puzzle is shown at the center of the screen. Similar to our second prototype, we use the BLOSUM62 matrix [Henikoff and Henikoff 1992a] for representing the relative mutation rates between the different amino acid types. However, we further redesigned the BLOSUM62-based color scheme in order to provide the player with an even better visual representation of the underlying amino acid

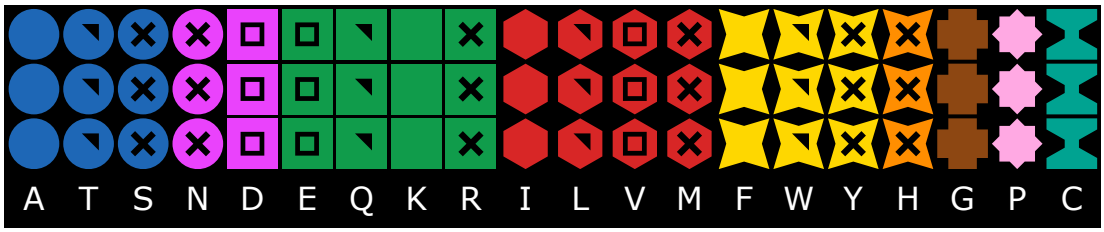


Figure 3.12: BLOSUM62 residue theme used as default theme in Bionigma. Substitutions between identically-colored residues are always rewarded with positive scores and are thus preferred. Additionally, similarities between differently-colored residues are represented by identical shape and texture (S vs. N). The highest rewards are received for substitutions between identical residues.

similarities encoded in BLOSUM62. Unlike our previous approaches, we visually encode these similarities in Bionigma using three different visual properties – color, shape, and texture. The resulting *residue theme* is shown in Figure 3.12.

Game tokens with identical colors represent amino acid groups that are guaranteed to receive positive or at least neutral substitution scores (zero) in BLOSUM62. For instance, the aliphatic amino acids I, L, M, and V are colored in red since substitutions between these acids are rewarded by positive BLOSUM62 scores. By aligning red game tokens, the player is thus guaranteed to improve its alignment score. The different shapes and textures are used in a similar manner. In most cases, identically-colored tokens also possess identical shapes to further emphasize their similarity, while the textures are used to indicate the unique acid type.

In some cases, however, the shape and/or texture properties are also used to represent similarities between game tokens (amino acids) that have different colors. For instance, the amino acids serine (S) and asparagine (N) are colored in blue and purple, respectively, but receive positive scores in BLOSUM62. Simply coloring S in purple would solve this issue but would wrongly yield a similarity between S and histidine (H). Additionally, the (visual) similarity between S and A and S and T would be destroyed. For this reason, we visually reflect the similarity between S and N by identical shape and texture.

According to BLOSUM62, substitutions between the amino acid types glycine (G), proline (P), and cysteine (C) and other amino acids are always penalized or receive a score of zero. Hence, the player only receives positive scores when aligning these types with themselves. To visually emphasize this behavior, we assigned unique shapes and colors to these three acid types.

Highlighting modes

To support the player in the alignment process, we implemented two different highlighting modes that can be activated on the player's demand using the buttons shown at the top left corner of the game interface. When activating the *similarity mode*, the player can move the mouse cursor over a specific game token to show similar tokens in the puzzle. This results in dissimilar tokens being shown with reduced size which visually emphasizes similar tokens that provide positive substitution scores. This principle is illustrated in Figure 3.13. Here, a token in the form of a yellow star with a cross texture was selected resulting in other

Figure 3.13: Bionigma's similarity mode for highlighting similar game tokens to the currently hovered token. Similar tokens are visually emphasized by reducing the size of dissimilar tokens.

star-shaped tokens being highlighted. This highlighting mode allows the user to easily detect similar game tokens and can be used, e.g., to reveal potential misalignments that could be improved.

The latter aspect is also addressed by our second highlighting mode called *bad token mode*. In this mode, the three worst scoring tokens types per column are visually emphasized by reducing the size of well-scoring game tokens. This mode is shown in Figure 3.14. It enables the player to directly assess the worst-performing tokens that could be re-aligned to further improve the alignment score. Besides these modes, the player can optionally highlight gaps by showing "gap tokens" as white dashes in the puzzle.

Special interactions

In addition to the aforementioned selection and movement interactions, we implemented two different special interactions based on the feedback received during our user studies. Both interactions can be triggered by clicking on the corresponding buttons shown at the lower left corner of the game interface. By clicking on the *align-left* or *align-right* buttons, the selected tokens are re-aligned to form a left or right justified alignment. This enables the player to quickly form a straight vertical alignment border without the need of manually moving individual tokens.

The second special interaction is an alignment mechanic we call *threading*. After triggering this alignment action, the player can select one token per row. By clicking on a column selector button of choice, the selected tokens are aligned in the player-specified column. This interaction is similar to one would thread single beads and pull the thread at both

Figure 3.14: Bionigma's bad token mode for highlighting potentially improvable alignment regions. This mode visually emphasizes the three worst-scoring tokens per column by reducing the size of the well-scoring tokens.

sides to form a straight line. An illustration of this mechanic is shown in Figure 3.15. Using this alignment action, the player can easily align similar tokens in specific columns, e.g., to produce an initial alignment as preparation for further refinements.

3.5.2 Evaluation - Game experience and usability

We evaluated our citizen science game approach Bionigma with respect to two different aspects. First, we assessed its capabilities to deliver a true game experience as well as its usability and the helpfulness of the tutorial. Our second goal was to evaluate the accuracy of the player refined sequence alignments. This part of the evaluation is presented later in Section 3.5.3.

Methods

In order to evaluate Bionigma with respect to the delivered game experience and its usability we performed another user study with a sample size of of 26 persons (9 women and 17 men; age: 9 to 56 years). All test persons were visitors at the public day of GameDays 2015[4]. They had no specific background in scientific fields such as biology or computer science.

For this test setting, we used real protein MSAs provided by the Bionigma server, i.e., subsets of a selection of Pfam seed alignments (version 27) [Finn et al. 2016]. These levels were parameterized with individual gap opening and extension penalties as well as level-specific scoring thresholds. Further details about the underlying MSAs are presented later in Section 3.5.3. The test persons were allowed to play the game without any restrictions including no maximum playing time and were encouraged to report their immediate game

[4] http://www.gamedays2015.de/, last accessed 24.10.2017

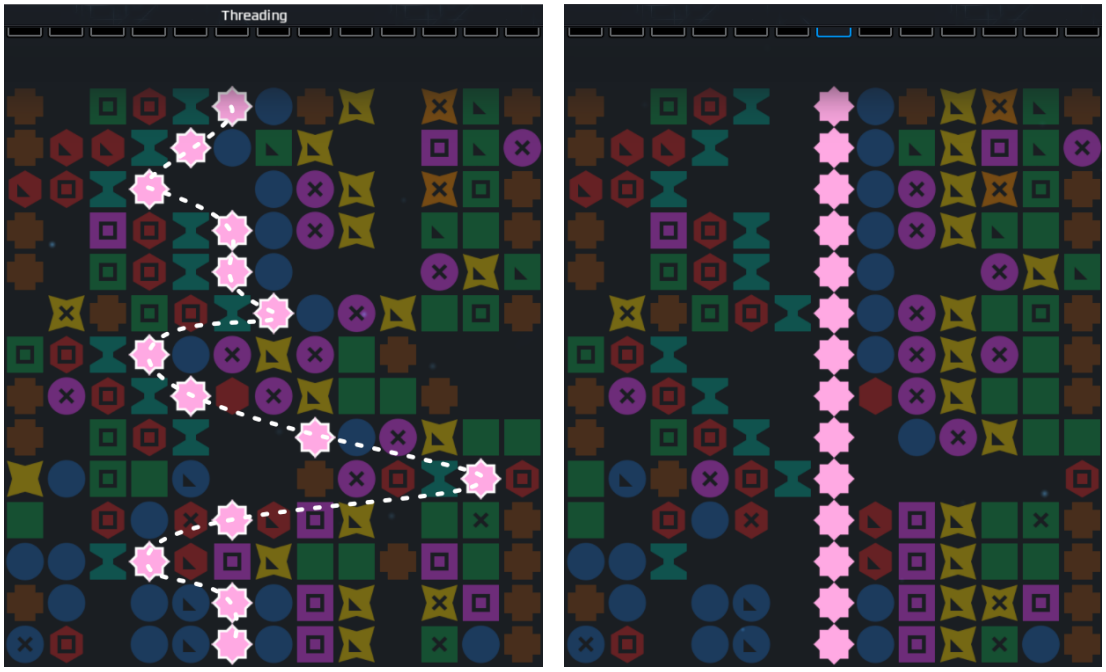


Figure 3.15: Example illustrating the threading alignment action. Left: The player can select one token per row. Hereby, a virtual thread is drawn through the selected tokens (dotted white line). Right: Clicking on a column selector of choice (blue rectangular button) triggers the alignment of the selected tokens in the corresponding column.

impressions while playing the game. After playing, the participants were asked to answer a questionnaire consisting of 24 items based on a Likert scale with five possible answers ranging from “not at all” to “extremely”.

The first 14 items referred to our modified version of the in-game experience questionnaire (iGEQ) [IJsselsteijn et al. 2008, Nacke 2009] (Table A.1) like in our previous studies. The remaining 10 items covered different aspects including usability, the usefulness of the tutorial and the provided BLOSUM62-based residue theme, and general game impressions. This part of the questionnaire is shown in (Table A.4).

Results - Game experience

The results of the iGEQ part of the questionnaire are illustrated in Figure 3.16. In these seven categories, Bionigma received predominantly positive ratings. This provides evidence that Bionigma delivers a true game experience to its players and is fun to play. In comparison to our previous prototypes, Bionigma even showed the highest immersion rating with a median of $\tilde{x} = 3$ and a mean of $\mu = 2.9$. The lower and upper quartiles for this category correspond to $Q_1 = 2$ and $Q_3 = 4$ also indicating that the majority of the players felt immersive during play.

In the flow category, Bionigma performed slightly better than its predecessor but still received only moderate ratings with median and mean values of $\tilde{x} = 2.5$ and $\mu = 2.35$, respectively. Notably, there is a large variance in the answers of the participants indicated by a lower

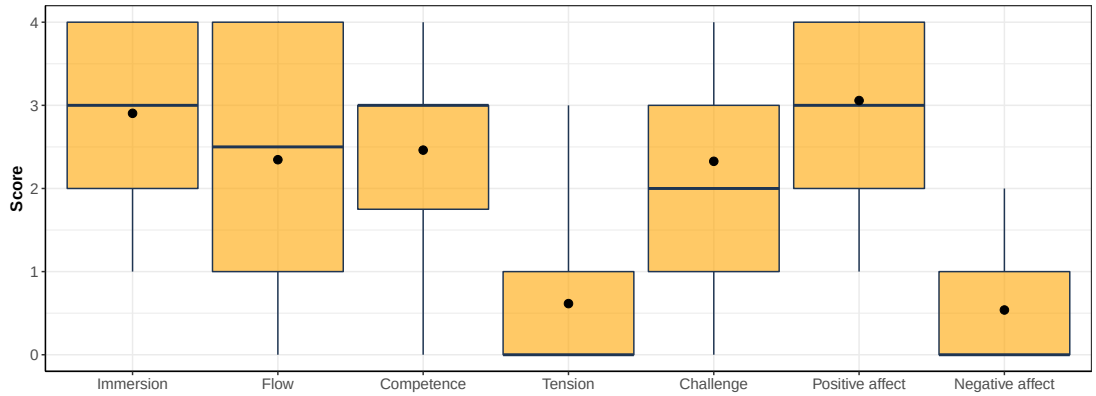


Figure 3.16: Box plot depicting the answer distributions of the different GEQ categories obtained in the user study of our citizen science game approach Bionigma.

quartile of only $Q_1 = 1$ and a high upper quartile of $Q_3 = 4$. While some participants reported that they completely lost their sense of time during play, other participants were apparently not affected by a considerable flow.

The majority of the participants reported a strong feeling of competence during play resulting in a median of $\tilde{x} = 3$. Still, 19% of all participants reported that they had problems in solving the levels and 31% of the test person did not feel skillful during play reducing the mean value to only $\mu = 2.46$. Interestingly, these issues had apparently no affect on the challenge category. Here, the players rated Bionigma overall moderately ($\tilde{x} = 2$, $\mu = 2.33$) indicating that the majority of the players considered the difficulty of the levels as well balanced.

Bionigma received very positive ratings in the category of negative affect ($\tilde{x} = 0$, $\mu = 0.54$). This demonstrates that the majority of the participants neither felt bored during play nor found the game tiresome. Likewise, the players experienced only a very low level of tension indicated by median and mean values of $\tilde{x} = 0$ and $\mu = 0.62$. Bionigma's overall good ratings in the category of positive affect ($\tilde{x} = 3$, $\mu = 3.06$) further shows that Bionigma is fun to play.

Results - Tutorial, usability and game mechanics

The results of the second part of our user study addressing the overall game impression and mechanics as well as the tutorial is shown in Figure 3.17. Also in this part, Bionigma received very positive feedback. The majority of the participants liked the game ($\tilde{x} = 3$, $\mu = 3.23$) and would have played longer if they had more time ($\tilde{x} = 3$, $\mu = 2.69$). Most of the participants were also interested in playing Bionigma in their leisure time ($\tilde{x} = 3$, $\mu = 2.58$).

Bionigma's tutorial also received overall positive ratings. Most players found the tutorial comprehensive ($\tilde{x} = 4$, $\mu = 3.5$), well-structured ($\tilde{x} = 3$, $\mu = 3.35$), and most importantly helpful ($\tilde{x} = 3$, $\mu = 3.31$). Additionally, the majority of the participants found the controls of Bionigma intuitive including the novel special interactions and were pleased with the precision of the interactions as indicated by median values of $\tilde{x} = 3$ and mean values of $\mu = 3.08$ and $\mu = 2.96$, respectively.

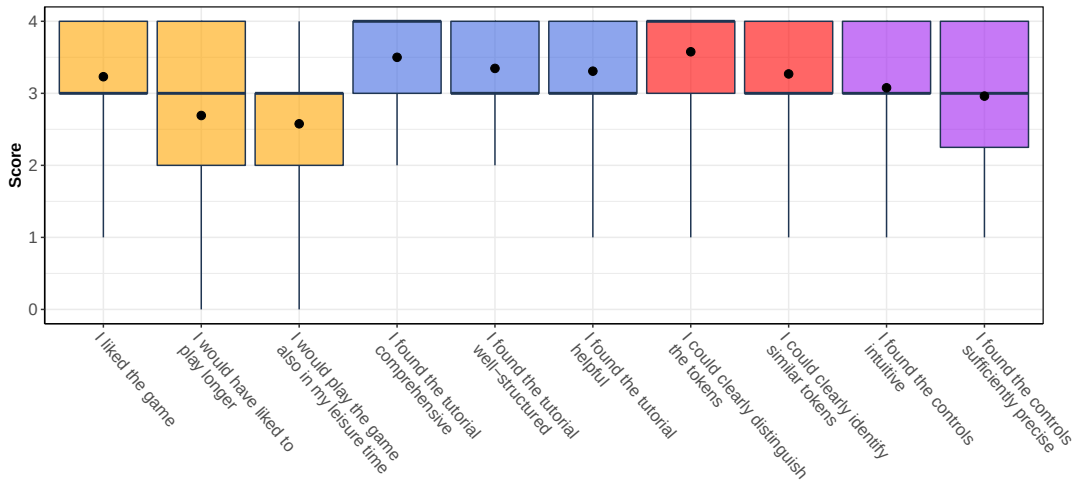


Figure 3.17: Box plot depicting the answer distributions obtained in the user study of Bionigma for the questions regarding the overall game impression (orange), the tutorial (blue), the residue theme (red), and the controls (purple).

Likewise, our novel residue theme using shape, color, and texture for representing similarities between different amino acid types received very positive ratings. Almost all players reported that they could clearly distinguish the different types of tokens ($\tilde{x} = 4$, $\mu = 3.58$) and also could successfully identify similar tokens ($\tilde{x} = 3$, $\mu = 3.27$).

Discussion

The results of our user study shows that our improved citizen game concept for the manual refinement of protein MSAs presented in the form of our digital game Bionigma actually works. The majority of the players had a lot of fun during play and are also interested in playing the game in their leisure time. Likewise, the game was considered to be visually appealing and attractive. The overall positive answers received in the GEQ Categories provide further evidence that Bionigma delivers a true game experience to the players.

Still, there appears to be room for improvements with respect to the ratings of the GEQ categories competence and flow. Roughly 31% of the players reported that they did not feel skillful during play. Interestingly, we did not find, however, a correlation between these answers and those obtained from the same persons regarding the challenge category. Thus, we relate the negative ratings received for the competence category primarily to the fact that the players played Bionigma for the first time and would probably feel more competent with increased experience.

The reported feeling of flow was higher on average ($\mu = 2.35$) compared to our second prototype ($\mu = 1.8$). Still, the answers received for this category varied strongly as indicated by the lower and upper quartiles of $Q_1 = 1$ and $Q_3 = 4$, respectively. Similar to our previous user study performed during the public day of GameDays 2014, a potential reason for this variance could be again external disturbances caused by the environment which negatively affected the concentration of some players. Unfortunately, we did not received additional information from the participants that could further explain the reasons for the only moderate flow ratings.

3.5.3 Evaluation - MSA accuracy

In the second of part of the evaluation, we analyzed the accuracy of the MSAs refined by the players of Bionigma. This includes the results obtained for the different levels played during the user study described before and those received by the players during Bionigma's beta phase.

Methods - Level data

For this evaluation, we implemented a set of 32 game levels in total with difficulty ratings ranging from "easy" to "difficult". These levels are based on subsets of real protein MSAs derived from Pfam seed alignments (version 27.0) [Finn et al. 2016]. Pfam seed alignments are manually curated by experts and thus can be considered as ground truth sequence alignments providing us with reliable MSA data.

The 32 levels were created by selecting rectangular regions from the original alignments that were subsequently re-aligned by randomly inserting gaps. Even though this destroys the original alignment constellation, it enables the players to start their own alignment process from a more unbiased starting point. This also avoids the need to generate the entire alignment from scratch. Additionally, we found that most players had more fun in playing largely unaligned puzzles rather than improving puzzles pre-aligned by experts. Our method thus also aims at producing levels that are fun to play.

We assigned each level a difficulty rating of either easy, normal, advanced, or difficult. This rating depends on the size of a level, its overall sequence similarity, and the chosen gap penalties. For most levels, we assigned a gap opening penalty of -10 and a gap extension penalty of -2. This is in concordance to often chosen penalty settings by other tools (e.g., BLAST [Altschul et al. 1990], FASTA [Pearson 1991]). Additionally, we created copies of existing levels and assigned them higher penalties to investigate the impact of different penalty heights. Two copies were assigned a gap opening and extension penalty of -35 and -2 and four copies received penalties of -45 and -2, respectively.

Methods - Alternative MSAs and quality measures

We analyzed the accuracy of the player-refined MSAs in comparison to the original Pfam seed alignments and alternative MSAs generated with the frequently used MSA program MUSCLE [Edgar 2004b] (Section 2.4.2). Here, we applied the same scoring parameters as in Bionigma, i.e., the BLOSUM62 matrix as substitution model and the aforementioned gap penalties. The accuracy of the different alignments was measured using the total SP score and the scoring model used in Bionigma, i.e., the sum of the total SP value and the total gap penalty.

Notably, we could not always compare the alternative alignments computed by MUSCLE and the Bionigma players with their original Pfam seed counterparts. Some Pfam seed alignments are too large to directly implement them as Bionigma levels. Therefore, we had to manually select subsets. The original alignments of these subsets were constructed with respect to the context of the entire sequence set. Since not all subsets properly reflect this context, we cannot always compare the original subset alignments with their alternatives generated by MUSCLE or the Bionigma community in a fair way. In these cases, we thus do not report accuracy scores for the original Pfam seed alignment subsets.

Results and discussion

The scoring results obtained for the alternative alignments of the Bionigma levels are shown in Figure 3.18 and Figure 3.19. The results for the total SP score are shown in the upper part and the lower half depicts the Bionigma score. If available, the scores for the original Pfam seed alignments are shown in red. The scores of the MUSCLE alignments are colored in yellow and the three highest-scoring MSAs generated by Bionigma players are depicted in blue. The individual scores obtained for each level by MUSCLE and the high-scoring player (the level master) and the scoring ratios between them are shown in Table A.5.

According to these results, the best Bionigma players always generated alignments with higher total SP and Bionigma scores than MUSCLE. Where comparable, these player alignments also received higher scores than the original Pfam seed alignments. Under the applied scoring models (total SP and Bionigma score), Bionigma players thus produced better alignments than MUSCLE and also the original Pfam seed alignments. The alignments created by the players of Bionigma contain, however, much more gaps on average than the MUSCLE alignments and the original Pfam seed MSAs. This effect can be observed in particular for larger levels with gap opening and extension penalties of -10 and -2.

From a biological and evolutionary perspective, some of the MSAs generated by the Bionigma community are thus doubtful even though they received high total SP and Bionigma scores. An example illustrating this observation is shown in Figure 3.20. Here, the player could achieve very high scores although he or she avoided almost all mismatches and introduced numerous gaps. Notably, empty columns in this example are only used by the player for structuring purposes and do not represent real gaps that are penalized.

The undesired effect of too many gaps can be related to the gap penalty model used in Bionigma. Gaps are penalized row-wise but substitutions are rated pairwise. Substitutions thus semi-quadratically influence the final score, while gaps only have a linear impact. For this reason, one has to apply high gap penalties to balance both terms. In particular, the penalties must be adjusted according to the number of sequences in the alignments. As shown by the player MSA depicted in Figure 3.20, we apparently selected too low penalties for some levels.

Nonetheless, we also obtained biologically and evolutionary reasonable MSAs from the Bionigma community. For example, Figure 3.21 shows two alternative alignments of the Bionigma level Usher-Protein-P1c using gap opening penalty of -35 and an extension penalty of -2. Shown at the top is the original alignment of the Pfam seed MSA subset. Interestingly, this MSA is identical to the MSA generated by MUSCLE and contains only terminal gaps. The highest-scoring alternative alignment created by a Bionigma players is shown at the bottom. Here, the Bionigma player decided to align the proline residues (pink stars) at the center and at the C-terminal region (right hand side) of the alignment.

This is interesting because proline is known to be a so-called “helix breaker”. Alpha helices are common secondary structure motifs of proteins. They are held together by hydrogen bonds. Proline’s molecular structure prevents, however, its amino group from participating in hydrogen bonding. The existence of a proline residue in a sequence may indicate the end of an alpha helix. From a structurally driven alignment perspective, it thus makes sense to align the proline residues in this Bionigma level.

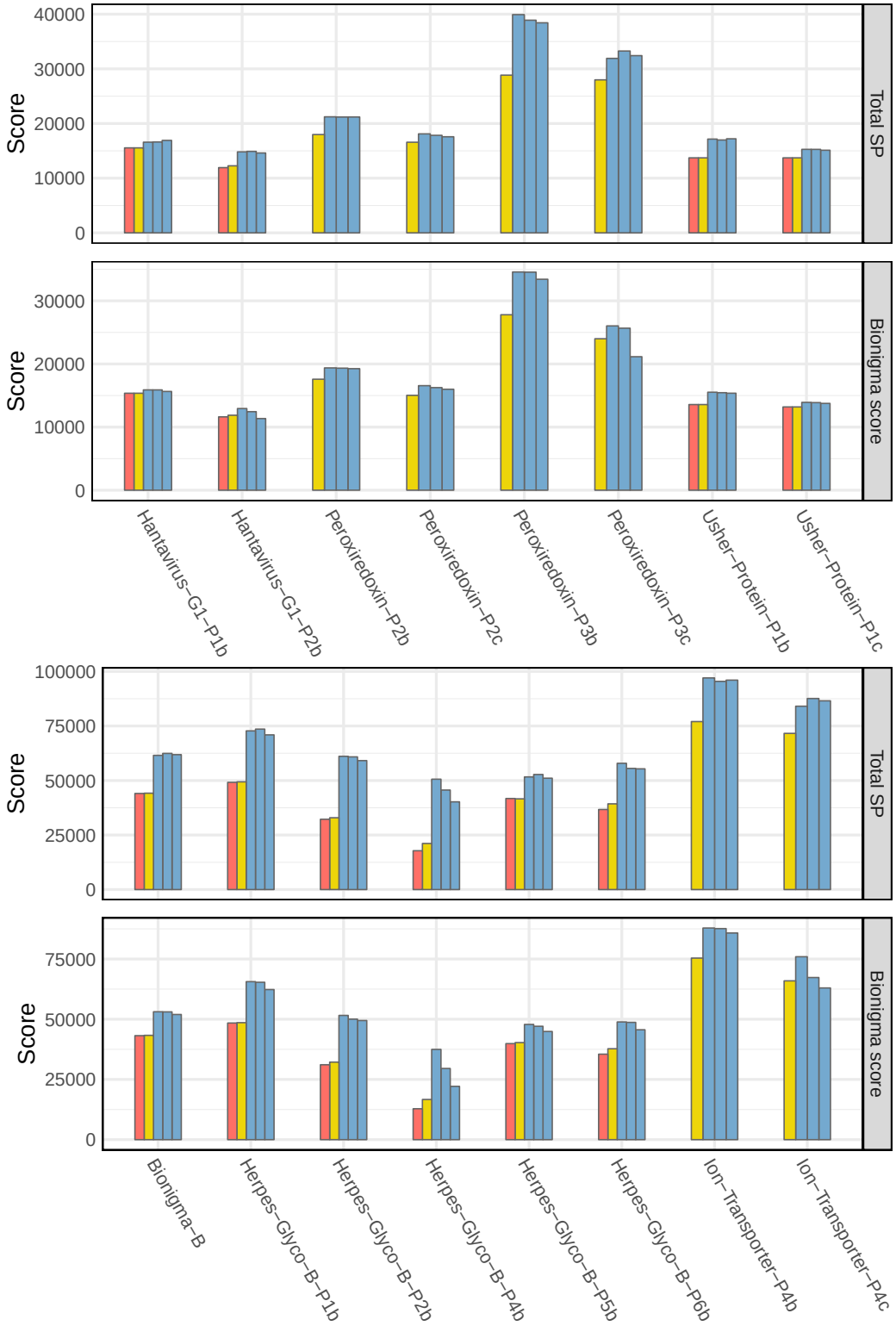


Figure 3.18: Scoring results obtained for the first 16 levels. The upper parts of the two shown level groups represent the total SP score, the lower parts the Bionigma score.

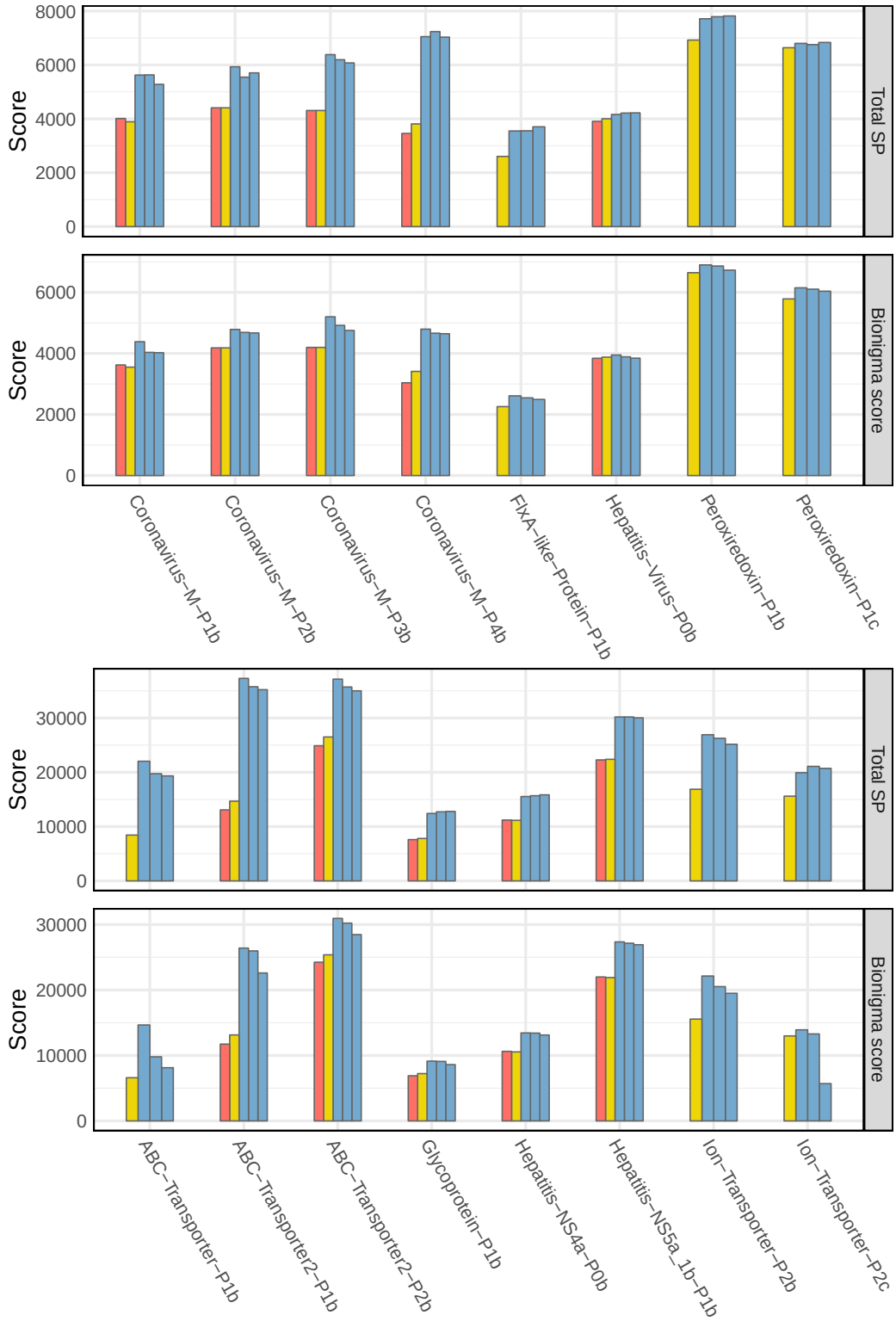


Figure 3.19: Scoring results obtained for the level numbers 17 to 32. The upper parts of the two shown level groups represent the total SP score, the lower parts the Bionigma score.

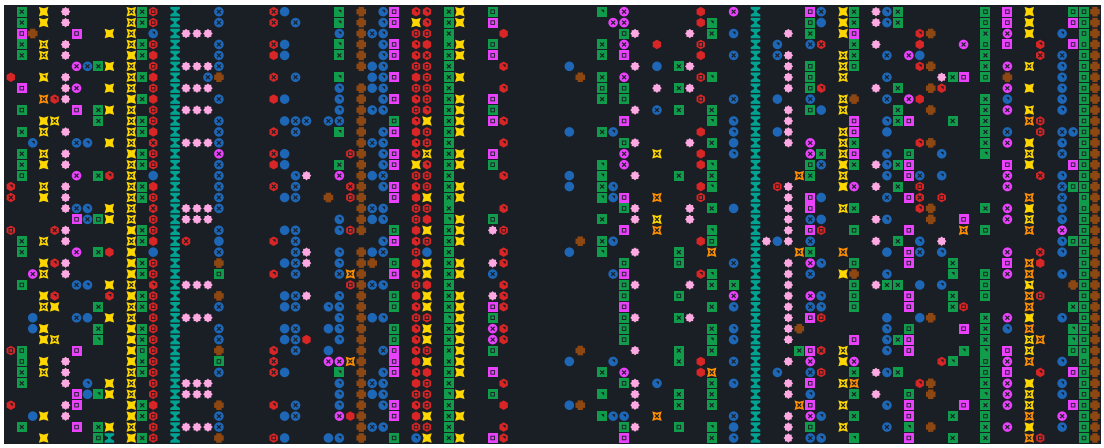


Figure 3.20: A player solution for the Bionigma level Herpes-Glyco-B-P1b using a gap opening penalty of −10 and an extension penalty of −2.

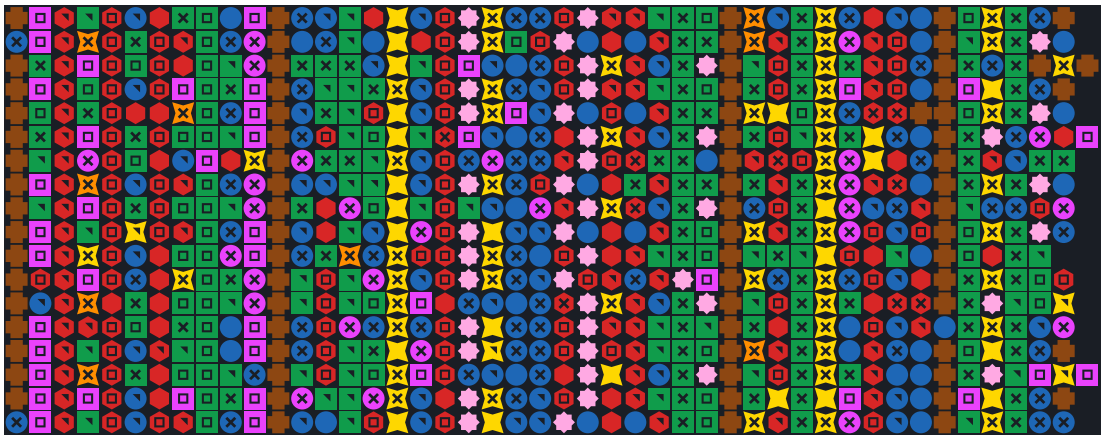
Figure 3.22 shows another example for the Bionigma level Ion-Transporter-P4c. The MUSCLE alignment is shown at the top and the highest-scoring player alignment is depicted at the bottom. The original Pfam seed alignment is not shown in this case because it cannot be compared in a fair way as outlined above. Overall, both alignments are relatively similar. The MUSCLE alignment contains generally fewer gaps than the player alignment but has more terminal gaps. The latter can be related to the reduced penalty that is applied by MUSCLE for terminal gaps. This apparently results, however, in misalignments at the C-terminal alignment region. In contrast, the Bionigma player aligned this region more reasonably.

Unlike MUSCLE, the player introduced several gaps of length one in the central region of the alignment. This is contrary to common knowledge that longer consecutive gaps are considered to be more likely than several single gaps at non-consecutive positions [Lesk 2013, p. 184]. In this case, however, adding these gaps substantially reduces the number of mismatches between very dissimilar residues. This also leads to significantly increased total SP scores. From this perspective, adding these gaps thus may be considered to be plausible.

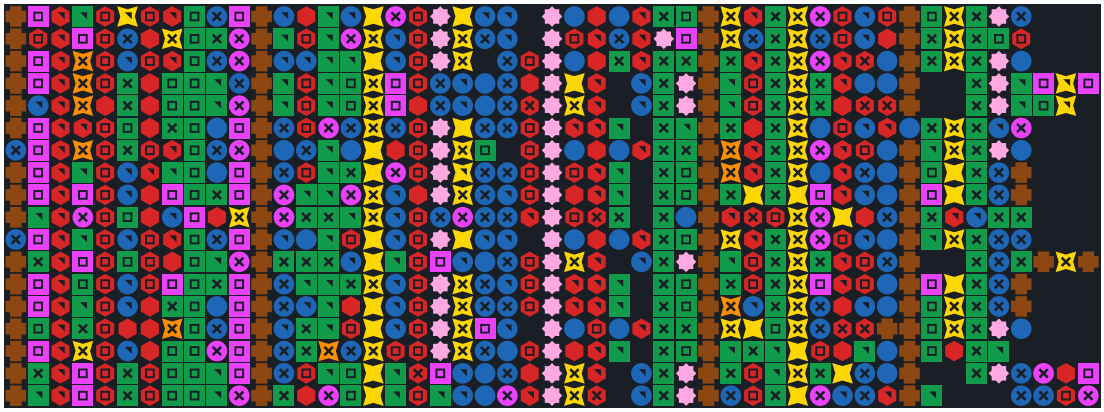
3.5.4 Conclusion

We presented Bionigma, an improved scientific discovery game approach for the manual refinement of protein sequence alignments. Bionigma abstracts the alignment problem in the form of a puzzle game where the player must align similar game tokens. In order to visually encode the similarities between the different amino acid types, we proposed a novel visualization theme based on the BLOSUM62 substitution scores [Henikoff and Henikoff 1992a] using different colors, shapes, and textures. Furthermore, we presented novel highlighting and interaction modes to support the players in the alignment process. These features in combination with a visually appealing game interface enable even non-experts – casual players – to successfully improve sequence alignments.

As shown by our user study, Bionigma is fun to play and delivers a true game experience to the players. The resulting player-refined MSAs also showed promising results. According to commonly used scoring models, Bionigma players produced better alignments than the alignment program MUSCLE [Edgar 2004b]. In view of biologically and evolutionary



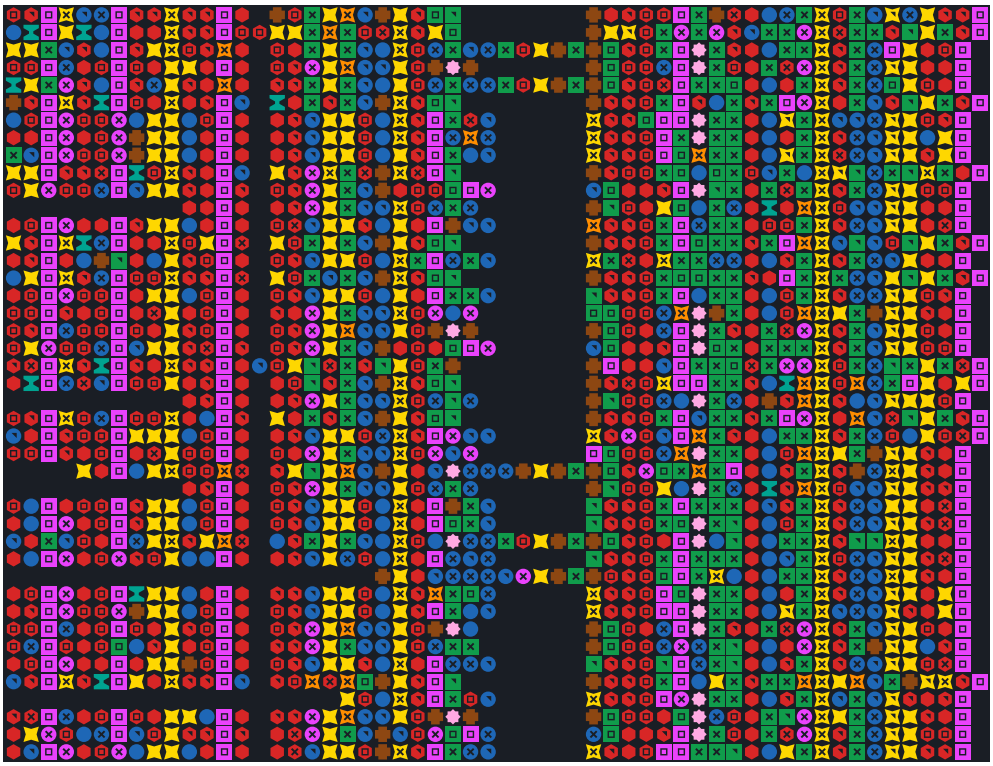
Original Pfam seed and MUSCLE alignment



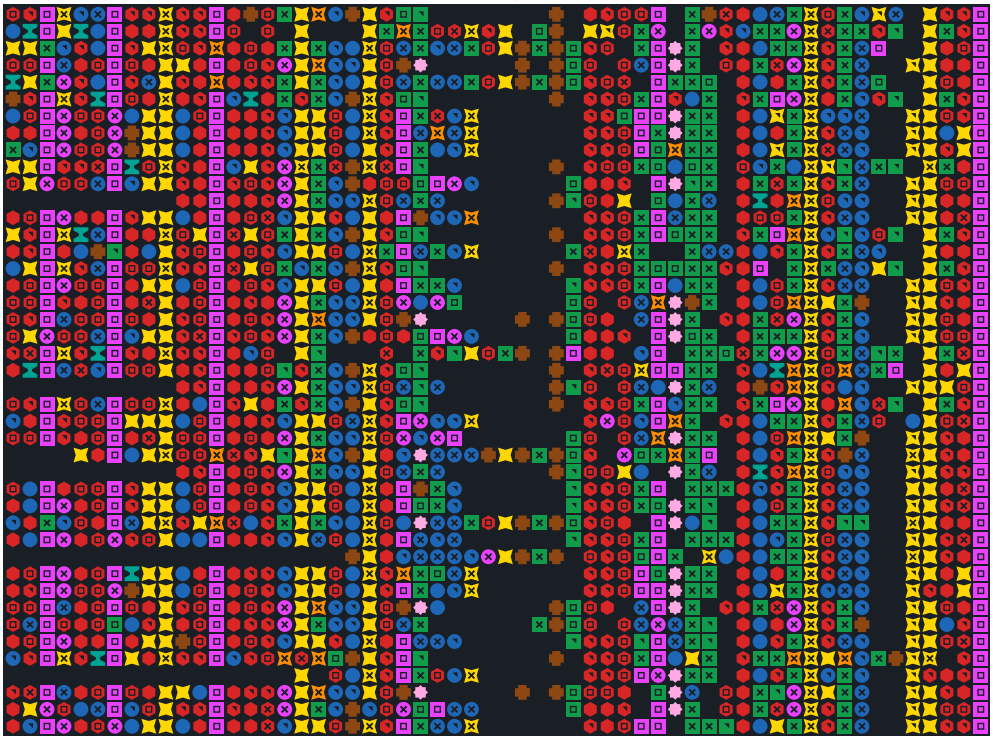
Best player alignment

Figure 3.21: Alternative alignments of the Bionigma level Usher-Protein-P1c.

correctness, we found that the accuracy of the obtained alignments strongly depends on the heights of the chosen gap penalties. For sufficiently large gap penalties, the players could, however, successfully produce plausible alignments.



MUSCLE alignment



Best player alignment

Figure 3.22: Two alternative alignments of the Bionigma level Ion-Transporter-P4c.

Chapter 4
Substitution matrices

4.1 Introduction

The computation of sequence alignments usually requires a particular scoring model which represents evolutionary substitution, insertion and deletion events. While the latter two events are normally encoded by specific gap penalty models as described in Section 2.1.2, substitution events are represented in the form of substitution matrices. Each matrix entry encodes the substitution event between two specific types of nucleotides or amino acids α_i and α_j. The values on the off-diagonal entries represent the relative mutation rates between α_i and α_j, i.e., the likelihood of α_i mutating to α_j in relation to independent evolution. The likelihood of preserving a particular nucleotide or amino acid is represented on the diagonal of the substitution matrix.

The computation of DNA sequence alignments is often based on simple substitution models [Lesk 2013, p. 184]. This includes, e.g., the identity matrix or a matrix that rates preserved pairs with +1 and substitutions with -1. The generation of a protein sequence alignment, however, usually requires a more complex model since substitution events between at least 20 different standard amino acids need to be encoded. In addition, some amino acid types are considered to mutate more likely into other types that have similar size or physicochemical properties [Lesk 2013, p. 184]. A protein substitution matrix thus has to represent these "similarities", especially when it is used for the alignment of distantly related proteins which may contain numerous point mutations.

Protein substitution matrices normally encode the relative mutation rates in the form of log-odds scores which are commonly derived by counting amino acid substitution frequencies in aligned sequence datasets. The unrounded log-odds score S_{α_i,α_j} for two amino acids α_i and α_j corresponds to the following equation:

$$S_{\alpha_i,\alpha_j} = \log_2 \frac{p(\alpha_i, \alpha_j)}{p(\alpha_i)p(\alpha_j)}$$

The term $p(\alpha_i, \alpha_j)$ represents the relative substitution frequency for the amino acids α_i and α_j in the aligned sequence dataset, i.e., the number of $\alpha_i\alpha_j$ pairings $n(\alpha_i, \alpha_j)$ in relation to the total number of observed substitutions $N = \sum_{\alpha_i,\alpha_j} n(\alpha_i, \alpha_j)$. The terms $p(\alpha_i)$ and $p(\alpha_j)$ correspond to the marginal probabilities for observing an amino acid α_i and α_j, respectively. They can be derived by counting the number of amino acids of type α_i and α_j in the dataset and relating these counts to the total number of amino acids. The final log-odds scores are often rounded and scaled to mitigate numerical problems.

Since the publication of the first commonly used protein substitution matrix by Dayhoff et al. [1978], a large number of different protein substitution matrices for several application purposes have been developed. For instance, some focus on providing a scoring model for very specific problems such as searching for proteins that contain similar transmembrane regions [Ng et al. 2000]. Other matrices [Dayhoff et al. 1978, Henikoff and Henikoff 1992a, Styczynski et al. 2008, Müller et al. 2002, Hess et al. 2016a, Keul et al. 2017] are used in a more generic context, e.g., when searching homologs of newly discovered protein sequences with unknown properties or when aligning distantly related proteins.

In the following section, we will present and discuss state-of-the-art protein substitution matrices which are commonly used for homologous sequence search and for the computation of multiple sequence alignments. Methodologies for measuring the performance of substitution matrices in the context of these tasks are subsequently described in Section 4.3. The remainder of this chapter presents our contributions in the research field of substitution matrices.

First, we describe an error correction for the popular BLOSUM algorithm [Henikoff and Henikoff 1992a] resulting in the new CorBLOSUM matrix series [Hess et al. 2016a]. These matrices deliver improved homology search results when compared to BLOSUM (Section 4.4). Second, we present a novel substitution matrix type called PFASUM [Keul et al. 2017] which outperforms commonly used state-of-the-art matrices for homologous sequence search and the computation of multiple sequence alignments (Section 4.5). Both substitution matrices and the corresponding evaluations were jointly developed and performed by Frank Keul and myself.

4.2 Related work

Over the last years, numerous protein substitution matrices have been developed which aim at different application scenarios in the context of homologous sequence search and MSA computation [Dayhoff et al. 1978, Gonnet et al. 1992, Jones et al. 1992, Henikoff and Henikoff 1992a, Ng et al. 2000, Kann et al. 2000, Müller and Vingron 2000, Müller et al. 2002, Styczynski et al. 2008, Song et al. 2015]. Due to the large number of different substitution matrices, we focus in this section on commonly used substitution matrices for homology search and MSA construction.

4.2.1 PAM - Point accepted mutation matrix

The PAM matrix series developed by Dayhoff et al. [1978] is the first set of commonly used protein substitution matrices in the historical context. It is still employed today, especially when analyzing closely related sequences, e.g., using popular homology search programs such as BLAST [Altschul et al. 1990] and FASTA [Pearson and Lipman 1988]. PAMn matrices model amino acid mutation probabilities for certain evolutionary distances n using Markov chain models, i.e., the number of point accepted mutations (PAM) between two sequences. The initial PAM1 matrix represents the evolutionary distance of 1% amino acid changes on average. It is derived from $1,572$ amino acid substitutions observed in a set of very closely related sequences. Markov chains for constructing PAM matrices for larger distances of n PAM can be obtained by simply multiplying the initial PAM1 matrix n times with itself.

Unlike other originating databases for substitution matrices, the dataset used by the PAM matrices is very small compared to the large protein databases available today. PAM matrices thus may be used for the detection of relationships between very closely related sequences but their capabilities in the context of the alignment of distantly related proteins is very limited [Henikoff and Henikoff 1993].

4.2.2 VTML - Variable time maximum likelihood matrix

The VTML substitution matrices [Müller and Vingron 2000, Müller et al. 2002] are another popular matrix series that is widely used for the computation of sequence alignments. These matrices were originally developed for a better detection of distantly related homologs. Still, it is also commonly used to compute high quality multiple sequence alignments [Edgar 2004b]. VTML matrices are constructed by iteratively estimating evolutionary distances and substitution rates from a set of pairwise sequence alignments using a maximum likelihood estimator. As initial rate matrix for this process, the VTML matrices uses Dayhoff's substitution model described above.

The pairwise alignments are obtained by randomly sampling two pre-aligned sequences from each protein family in the SYSTERS database [Krause and Vingron 1998]. This dataset is much larger and more diverse compared to the data basis used by the PAM matrices. This allows VTML matrices to provide a more reliable detection of remote homologs. Only pairwise alignments are considered, however, to prevent bias from oversampling. The covered sequence space by these alignments is still rather small compared to the sequence space available today.

4.2.3 BLOSUM - Blocks substitution matrix

One of the most popular and commonly used substitution matrix types is the BLOSUM matrix series (**BLO**cks **SU**bsitution **M**atrix) by Henikoff and Henikoff [1992] with BLOSUM62 being the most prominent matrix. As described in Section 4.1, BLOSUM matrices represent amino acid mutation rates in the form of rounded and scaled log-odds scores. The underlying joint probabilities $p(\alpha_i, \alpha_j)$ and marginals $p(\alpha_i)$ and $p(\alpha_j)$ for observing the amino acid types α_i and α_j are derived from aligned and conserved amino acid blocks stored in the BLOCKS 5.0 database [Henikoff and Henikoff 1991]. Each block represents a set of related sequence segments of equal length λ and is processed separately in the matrix construction. Afterwards, the substitution and amino acid frequencies observed for each block are merged into a single matrix and subsequently transformed into the final log-odds scores.

In contrast to the PAM matrix series, BLOSUM matrices do not directly model specific evolutionary distances using Markov chain models and the number of point accepted mutations. Instead, they re-weight the observed substitutions on basis of relative sequence similarities. This mitigates the potential bias of overfitting that can occur when counting amino acid changes between highly conserved sequences.

Before counting the amino acid changes $n(\alpha_i, \alpha_j)$ in a block, the sequences inside the block are clustered based on their relative similarity Φ and a predefined threshold t. For each two aligned sequences A and B in the block, the relative similarity $\Phi(A, B)$ is calculated by counting the number of aligned amino acid pairs that share the same amino acid type and normalizing this count by the block width λ. The obtained similarity $\Phi(A, B)$ is then compared with the preset similarity threshold t in order to determine the cluster membership of A and

B. First, each sequence is assigned to a separate cluster. The clusters are then iteratively merged until no changes are detected. Hereby, two clusters c_x and c_y are merged into a single cluster if the similarity $\Phi(A,B)$ of at least one sequence A in c_x and one sequence B in c_y is greater or equal than the preset threshold t, i.e., $\Phi(A,B) \geq t$ with $A \in c_x$ and $B \in c_y$.

When processing the aligned amino acid pairs, substitutions between sequences that belong to the same cluster are completely ignored. For all other substitutions between two sequences A and B that belong to different clusters, each observed pair is weighted by the product of the corresponding cluster sizes. In other words, each cluster is considered as a single sequence in counting substitutions. For example, if A belongs to the cluster c_x with a size of $|c_x| = 3$ and B to another cluster c_y of size $|c_y| = 4$, then each observed pair of the amino acids α_i and α_j is counted as $\frac{1}{|c_x| \cdot |c_y|} = \frac{1}{12}$. The number suffix in the BLOSUM matrix names indicate the similarity threshold used in the construction process. For instance, the BLOSUM62 matrix was created using a similarity threshold of 62%, i.e., $t = 0.62$.

BLOSUM matrices and especially the BLOSUM62 matrix are the de facto standard for homology search. They are also often employed as substitution model for the computation of multiple sequence alignments, for instance in `ClustalW2` [Larkin et al. 2007] and `MAFFT` [Katoh et al. 2002]. However, their underlying data basis in the form of the BLOCKS 5.0 database is rather small and dated compared to the large protein databases available today. This may limit their capabilities as several studies suggested that larger originating datasets can lead to significantly better performing substitution matrices [Price et al. 2005, Hess et al. 2016a]. Another important issue with the BLOSUM matrices is that many popular programs still use the matrix versions calculated with the initial BLOSUM program implemented in 1992, even though these versions are known to be substantially biased due to implementation errors [Styczynski et al. 2008, Hess et al. 2016a].

4.2.4 RBLOSUM - Revised BLOSUM matrix

In 2008, Styczynski et al. [2008] identified an implementation error in the original BLOSUM source code which affects the weighting procedure employed in the substitution frequency counting step. The original BLOSUM implementation only considered the size of the first cluster for the re-weighting of the observed substitution frequencies instead of the product of both cluster sizes as originally intended [Henikoff and Henikoff 1992a]. The correction of this implementation error led to substantially different matrix compositions denoted as RBLOSUM matrices (Revised BLOSUM). However, the usage of these corrected RBLOSUM matrices for homologous sequence search was reported to produce less accurate search results compared to the original BLOSUM matrices.

Interestingly, the RBLOSUM matrices are barely noticed in the scientific community even though they are closer to the substitution matrices originally intended by Henikoff and Henikoff. As of Google Scholar[1], the RBLOSUM paper was cited only 77 times since its publication in 2008, while the original BLOSUM paper received 3010 citations during the same time period. As a result, the actually incorrect BLOSUM matrices are still widely used and affect homologous sequence search results as well as MSAs.

[1] https://scholar.google.de, last accessed 28.09.2017.

Notably, a similar effect can be observed for the usage of sequence alignment tools where most people still tend to use programs that are not as accurate as others [Chatzou et al. 2016]. According to Chatzou et al. [2016], an explanation for this effect could be the potential "existence of a strong methodological inertia within the biological community, where tool usage tends to snowball through protocol recycling".

4.2.5 Summary

Over the last years, several substitution matrices have been developed to address different application scenarios. However, the arguably most popular matrices are still the PAM, BLOSUM, and VTML matrix series. They are commonly suggested and used as default parameter for state-of-the-art homology search programs [Altschul et al. 1990, Pearson 1991] and MSA tools [Edgar 2004b, Katoh et al. 2002]. While employing these matrices for both tasks usually produces reasonable results, their capabilities may still be limited. First, the de facto standard matrices for homologous sequence search – the BLOSUM matrices – are substantially biased due to implementation errors [Styczynski et al. 2008, Hess et al. 2016a]. Second, all matrices were derived from filtered and rather small datasets compared to the modern databases available today. This may limit their usefulness as shown by previous studies [Price et al. 2005, Hess et al. 2016a]. To overcome these limitations, we propose a further error correction of the BLOSUM algorithm as well as present a novel substitution matrix derived from manually curated structural alignments covering the currently known sequence space.

4.3 Methods

This section provides detailed information about assessing the performance of substitution matrices for the tasks of homologous sequences search and MSA computation. First, we describe the state-of-the-art methodology used for homology search performance benchmarks and assessing the statistical significance of the obtained results. The second part presents methods to measure the capabilities of substitution matrices for the generation of sequence alignments based on MSA benchmark datasets, corresponding reference alignments, and comparison measures.

4.3.1 Measuring homology search performance

One of the most important applications for protein substitution matrices is the search for homologous sequences given a particular query protein sequence. The state-of-the-art approach for assessing the homology search performance of substitution matrices is to perform homologous sequence search on a standardized database with known sequence relations [Brenner et al. 1998, Brenner et al. 2000]. In this context, the ASTRAL database [Brenner et al. 2000, Chandonia et al. 2004] serves as a gold standard for the assessment of homology search performance and parameter selection [Brenner et al. 1998, Green and Brenner 2002, Price et al. 2005, Styczynski et al. 2008]. The database itself is a subset of the SCOP/SCOPe databases [Murzin et al. 1995, Fox et al. 2014] and consists of structural alignments based on the hand-curated SCOP classification [Brenner et al. 2000, Chandonia et al. 2004].

Typically, all sequences of the ASTRAL database are searched against the entire database by employing homology search programs such as BLAST [Altschul et al. 1990] or SSEARCH [Pearson 1991]. Additionally, these program are parameterized with the substitution matrices that should be analyzed and varying gap penalties. The variation of the gap penalties allows to address the bias from suboptimal gap penalty settings. For this benchmarking purpose, the usage of SSEARCH is often favored over BLAST since SSEARCH is reported to produce more accurate results [Henikoff and Henikoff 1992a, Green and Brenner 2002, Styczynski et al. 2008]. Additionally, BLAST does not directly support the usage of arbitrary substitution matrices since BLAST additionally requires matrix and gap penalty model dependent alignment score statistics in order to compute *E*-values (Section 2.3). After running the homologous sequence search with the preferred parameter settings, a list of found homologs is obtained for each query sequence usually ordered by the reported *E*-values. The known sequence relations between a query sequence and its search results can then be used to determine whether the chosen parameters led to correct search results.

The coverage measure

A well established measure for this benchmark method is the coverage measure Q at a given errors per query value (epq) [Brenner et al. 1998]. Q represents the fraction of true positives found in the search results after applying an *E*-value threshold filtering based on the epq measure [Green and Brenner 2002, Price et al. 2005]. The classification of found homologs into true and false positives depends on the SCOPe sequence superfamily annotations. Typically, the maximum number of errors per query is set to 0.01 epq [Brenner et al. 1998, Green and Brenner 2002, Price et al. 2005, Styczynski et al. 2008]. This corresponds to a maximum of one allowed false positive relation identified per 100 queries on average for the entire database. For example, the search results obtained for the entire ASTRAL40 database (version 1.69) with its 7,290 sequences are filtered to contain no more than 72 false positives in total.

Since the different superfamily sizes found in the ASTRAL databases can result in potential bias when counting the number of true and false positives, the quadratic normalized coverage $\mathrm{Q}_{\mathrm{quad}}$ is often used as the average of true positive relations found per superfamily [Price et al. 2005]:

$$\mathrm{Q}_{\mathrm{quad}} = \frac{1}{S} \sum_{i=1}^{S} \frac{t_{\mathrm{i}}}{(s_i^2 - s_i)} \tag{4.1}$$

Here, t_i is the number of true positive relations found for a superfamily i with s_i sequences. S is the number of superfamilies in the database.

Concerted Bayesian bootstrapping

As the coverage measure strongly depends on the composition of the search database, the significance of the results can be estimated by Concerted Bayesian bootstrapping [Green and Brenner 2002]. This method effectively analyzes the influence of slight changes in the

database composition on the resulting coverage values. Applying the quadratic coverage normalization to the Concerted Bayesian bootstrapping yields the following equations for a single bootstrap:

$$\widehat{\mathrm{Q}}_i = \sum_{j=1}^{s_i} \sum_{m=1}^{N_j} \delta(\theta_j, \theta_m) w_j w_m \tag{4.2a}$$

$$\mathrm{W}_i = \sum_{k=1}^{s_i} \sum_{l=1}^{s_i} w_k w_l - \sum_{k=1}^{s_i} (w_k)^2 \tag{4.2b}$$

$$\widehat{\mathrm{Q}}_{\mathrm{quad}} = \frac{1}{S} \sum_{i=1}^{S} \frac{\widehat{\mathrm{Q}}_i}{\mathrm{W}_i} \tag{4.2c}$$

In Equation 4.2a, w_j represents the weight of the j-th query sequence of superfamily i drawn from a Dirichlet distribution. θ_j represents its superfamily annotation. Likewise, θ_m denotes the superfamily of the m-th query results for the j-th sequence with the weight w_m. $\delta(\theta_j, \theta_m)$ is the Kronecker delta, returning 1 if θ_j and θ_m are equal, i.e., if both sequences are members of the same superfamily, and zero otherwise. N_j is the number of homologs found for the query sequence and s_i denotes the sequence count of the i-th superfamily.

Thus, Equation 4.2a describes the unnormalized coverage for the i-th superfamily, i.e., all found "true positive" relations. Equation 4.2b is the quadratic normalization for the i-th superfamily, i.e., all possible positive interactions for the i-th superfamily. Summing over all relative coverages for the S-numbered superfamilies (Equation 4.2c) returns the quadratic normalized coverage for a single bootstrap.

The significance of the coverage difference of two matrix/gap combinations is tested by calculating a Z-score from a two-sample parametric means test using the variance from the two corresponding bootstrap distributions [Green and Brenner 2002]. Hereby, the Z-score measures the significance of the difference of the two underlying distributions (Equation 4.3).

$$Z_{p,q} = \frac{\bar{\mathrm{Q}}_p - \bar{\mathrm{Q}}_q}{\sqrt{\frac{\sigma_p^2 + \sigma_q^2}{N}}} \tag{4.3}$$

For two different matrix/gap combinations p and q, $\bar{\mathrm{Q}}_p$ and $\bar{\mathrm{Q}}_q$ represent the mean of the bootstrap coverages calculated for the p-th and q-th matrix/gap combinations at an errors per query (epq) of 0.01. σ_p^2 and σ_q^2 correspond to the variance of the underlying bootstrap coverage distributions. N represents the number of bootstrap rounds.

Coverage-based benchmark tools

A widely used toolkit [Price et al. 2005, Styczynski et al. 2008, Song et al. 2015] to calculate the coverage measure from SSEARCH results is the PSCE toolkit by Green and Brenner [2002]. However, this toolkit has limitations in terms of computation performance when processing

large amounts of SSEARCH results. In our studies, we thus used our own performance-optimized reimplementation of the PSCE toolkit called CoverageCalculator [Hess et al. 2016a].

The coverage calculation as implemented in CoverageCalculator considers a search result as a true positive relation if the superfamily annotations, as provided by the ASTRAL database, are identical for the query and the reported sequence. The *E*-value threshold for the filtering is selected adaptively, depending on the average number of false positive relations remaining in all search results after applying the threshold. A search result is considered a false positive relation if its superfamily annotation does not match the annotation of the query sequence. This is contrary to the PSCE toolkit, where search results with different superfamily but same fold annotation are ignored in the coverage calculation since their evolutionary relationship is unknown. Hence, our CoverageCalculator takes all reported results into account and thus, is not overestimating the "real" coverage by skipping unknown but real false positive relations within the same fold. Since the true evolutionary relationship between the superfamilies is not known, this may underestimate the "real" coverage, but consistently assumes that all superfamilies are not related. In other words, the coverages reported by CoverageCalculator represent a lower bound for the substitution matrix performance.

4.3.2 Measuring multiple sequence alignment performance

Another important field of application in computation biology for the use of protein substitution matrices is the computation of multiple sequence alignments. Similar to the state-of-the-art approach for measuring the homologous sequence search performance of substitution matrices, their capabilities in the context of MSA computation can also be assessed using standardized MSA benchmark datasets and specific MSA comparison measures (Section 2.5).

As outlined in Section 2.5.1, MSA benchmark datasets (BAliBASE 3.0 [Thompson et al. 2005], OXBench [Raghava et al. 2003], SABmark 1.65 [Van Walle et al. 2005]) usually provide specific sets of sequences and corresponding reference alignments which are often designed to represent specific alignment tasks. This includes, e.g., the alignment of structurally related sequences (BAliBASE 3.0) or very dissimilar sequences (SABmark 1.65).

For each set of sequences, a sequence alignment can be calculated using the substitution matrices that should be assessed in combination with a particular alignment algorithm and accompanying parameters such as varying gap penalties. The obtained “alternative” alignments can then be compared with the reference alignment by employing a particular comparison measure to quantify the differences between both alignments (Section 2.5.2). This effectively allows to measure the accuracy of the generated alignment and thus also implicitly analyzes the impact of the chosen substitution matrix on the resulting MSA.

A convenient method for performing this sort of analysis is given in the form of the benchmark collection bench [Edgar 2009a] and the program qscore [Edgar 2009b]. The benchmark collection bench consists of several state-of-the-art MSA benchmark datasets such as BAliBASE 3.0 [Thompson et al. 2005], OXBench [Raghava et al. 2003], SABmark 1.65 [Van Walle et al. 2005], while qscore provides an implementation of widely used MSA comparison measures such as the identically named *q-score* or the *Modeler score* by Sauder et al. [2000]. We use a subset of this collection and the qscore program to assess the capabilities of our novel PFASUM substitution matrices for the task of MSA construction.

4.4 CorBLOSUM substitution matrices

4.4.1 Introduction

As outlined in Section 4.2.4, Styczynski et al. [2008] discovered miscalculations in the clustering step of the popular BLOSUM matrix computation. Still, their revised RBLOSUM64 matrix performed significantly worse than the inaccurate original BLOSUM62 matrix. However, Styczynski et al. did not evaluate their correction in combination with other originating databases. Also, they focused in their study only on a single RBLOSUM matrix and a specific ASTRAL release (ASTRAL 1.69) as benchmark.

These limitations and the ever increasing coverage of the protein sequence space inspired us to re-evaluate the impact of the RBLOSUM correction on the resulting substitution matrices and their homology search performance. For the construction of the substitution matrices, we chose three different versions of the BLOCKS database [Henikoff and Henikoff 1991] as originating datasets. The BLOCKS 5.0 database represents the initial dataset used for the publication of the BLOSUM [Henikoff and Henikoff 1992a] and RBLOSUM matrices [Styczynski et al. 2008]. The BLOCKS 13+ database covers a larger sequence space and was reported to produce better performing BLOSUM matrices than those created with BLOCKS 5.0 [Price et al. 2005]. BLOCKS 14.3 represents the latest BLOCKS release from April 2007. This release spans the largest sequence space available in BLOCKS and represents a more conserved starting point for the parametrization of evolutionary models such as substitution matrices.

Based on the corrections presented by Styczynski et al. we modified the original BLOSUM code [Henikoff and Henikoff 1992b] to derive BLOSUM and RBLOSUM matrices from the aforementioned datasets. Thereby, we noticed an additional inaccuracy in the sequence clustering step (Section 4.2.3). This coding problem affects cluster memberships of sequences and necessitates modifications to both the original BLOSUM and the RBLOSUM variant. In short, the published code uses an inaccurate integer based threshold in the sequence clustering step so that sequences may be assigned to a particular cluster even though they do not meet the user-specified clustering threshold. While – on the surface – the induced inaccuracies appear to be minuscule, the resulting substitution matrix entries are systematically biased away from the actual conservation tendency intended by Henikoff et al. [Henikoff and Henikoff 1992a].

A correction of this inaccuracy in combination with the problems reported earlier by Styczynski et al. [2008] prompted us to derive a new substitution matrix series called CorBLOSUM [Hess et al. 2016a]. In the following subsections, we first provide a detailed description of this inaccuracy and analyze its impact on a theoretical basis. The remainder of this section discusses the compositional differences of the substitution matrices constructed by the different algorithms (BLOSUM, RBLOSUM, and CorBLOSUM) and different originating databases (BLOCKS 5.0.0, BLOCKS 13+, and BLOCKS 14.3). An exhaustive analysis of the matrices' capabilities for homologous sequence search is presented later in Section 4.4.3. Our analysis demonstrates that fixing small coding errors results in substantially different CorBLOSUM matrices which beneficially influences homology search performance in comparison to the original matrix.

```
    // Threshold calculation
680 int threshold = (int)(Cluster*(Block.width))/100;
    ...
    // Clustering decision
728 if (pairs[px].score >= threshold){
      // Cluster sequences
      ...
    }
```

Listing 4.1: Threshold calculation and clustering decision adapted from lines `680` and `728` of the original `blosum.c` file [Henikoff and Henikoff 1992b].

```
    // Corrected threshold calculation
680 float threshold = (float)(Cluster*(Block.width))/100f;
    ...
```

Listing 4.2: The floating point threshold calculation used in the CorBLOSUM computation.

4.4.2 Algorithm

As mentioned in Section 4.2.3, the BLOSUM algorithm employs a similarity-based sequence clustering step to mitigate the potential bias of overfitting when counting substitution frequencies in highly conserved amino acid blocks. In order to determine the cluster memberships of the sequences in a particular block, the original BLOSUM code performs two steps (Listing 4.1):

1. An integer clustering (`threshold`) is computed based on a user specified similarity value (e.g., 62% for BLOSUM62) defined in `Cluster` and the width of the currently processed block defined in `Block.width` (Listing 4.1, line 680).

2. The similarity score `pairs[px].score` calculated for each sequence pair `px`, i.e., the number of identical residues, is compared with the integer clustering `threshold`. If the score `pairs[px].score` is at least as high as the `threshold`, the sequences in `px` are assigned to the same cluster (Listing 4.1, line 728).

However, the usage of this integer threshold in the original BLOSUM code effectively truncates the *real* floating point threshold. This can lead to an inaccurate clustering decision in the subsequent clustering procedure. The following example illustrates this effect. At a block length of 93 amino acids, e.g., a minimum sequence similarity of 62% — corresponding to similarity value used to generate the BLOSUM62 substitution matrix — leads to a *real* similarity threshold of 57.66 identical residues. Thus, at least 57.66 identical amino acids have to be observed between two sequences in order to merge them into a cluster. In the original implementation, this value is truncated to 57 identical residues. In fact, this corresponds to an effective clustering value of just 61.29% which was not intended by the user and may result in mistakenly clustered sequences.

To avoid this inaccuracy, we use a floating point clustering threshold as shown in Listing 4.2. This modification in combination with the correction proposed by [Styczynski et al. 2008] results in the new CorBLOSUM algorithm and the corresponding CorBLOSUM matrix series.

4.4.3 Evaluation

We evaluated our CorBLOSUM construction method in comparison to the original BLOSUM and RBLOSUM algorithms in three different scenarios. First, we analyzed the impact of the clustering inaccuracy in the BLOSUM and RBLOSUM code from a theoretical point of view. Second, we compared the compositional differences of the three different matrix variants based on identical clustering thresholds and on the basis of similar relative entropy levels. While the former analysis demonstrates the impact of the different BLOSUM-type algorithms on the resulting matrix entries, the latter analyzes compositional differences between matrices with comparable capabilities for homologous sequence search [Altschul 1991]. In the last scenario, we examined the homologous sequence search performance of the three matrix variants on different ASTRAL databases with varying sequence similarity.

As the magnitude of both error corrections is influenced by the database composition and as newer BLOCKS releases are reported to produce better performing matrices [Price et al. 2005], we investigated matrices derived from three different originating databases in all scenarios: BLOCKS 5, BLOCKS 13+ and BLOCKS 14.3. The BLOCKS 5 database is the database used for the publication of the BLOSUM [Henikoff and Henikoff 1992a] and RBLOSUM matrices [Styczynski et al. 2008]. The BLOCKS 13+ covers a larger sequence space and was reported to produce better performing matrices than those created with BLOCKS 5 [Price et al. 2005]. BLOCKS 14.3 represents the largest and latest BLOCKS release. We added the labels 5.0, 13+ and 14.3 as subscripts to the matrix names to distinguish from which BLOCKS version a particular matrix is derived.

Theoretical error impact

On the surface, the above mentioned inaccuracy seems to be minuscule. However, when investigating this inaccuracy in detail with regard to different BLOCKS databases and underlying block lengths, a systematic bias can be identified. In order to quantify the effect of the integer type cast for a given similarity value K, we calculated the threshold difference ΔT_{rel} between the correct clustering threshold T and the threshold used in the original BLOSUM code $\widehat{T}$ in relation to different block lengths l_{block}:

$$\Delta T_{\text{rel}} = \frac{T - \widehat{T}}{l_{\text{block}}} = \frac{1}{l_{\text{block}}} * \left(\frac{K \cdot l_{\text{block}}}{100} - \left\lfloor \frac{K \cdot l_{\text{block}}}{100} \right\rfloor \right)$$

Figure 4.1 shows the impact of the theoretical error ΔT_{rel} as a function of the increasing block length and a similarity value of $K = 62\%$, i.e., the similarity value used for the construction of the popular BLOSUM62 matrix. The bar charts shown on the upper panels depict the number of blocks of a given length that can be found in the BLOCKS 5.0 and BLOCKS 14.3 databases. The former database corresponds to the BLOCKS version used for the original BLOSUM matrices containing 27,102 sequences. The latter dataset represents the last BLOCKS release. It contains two orders of magnitude more sequences than BLOCKS 5.0 (6,739,916 entries) which results in more incorrectly clustered sequences.

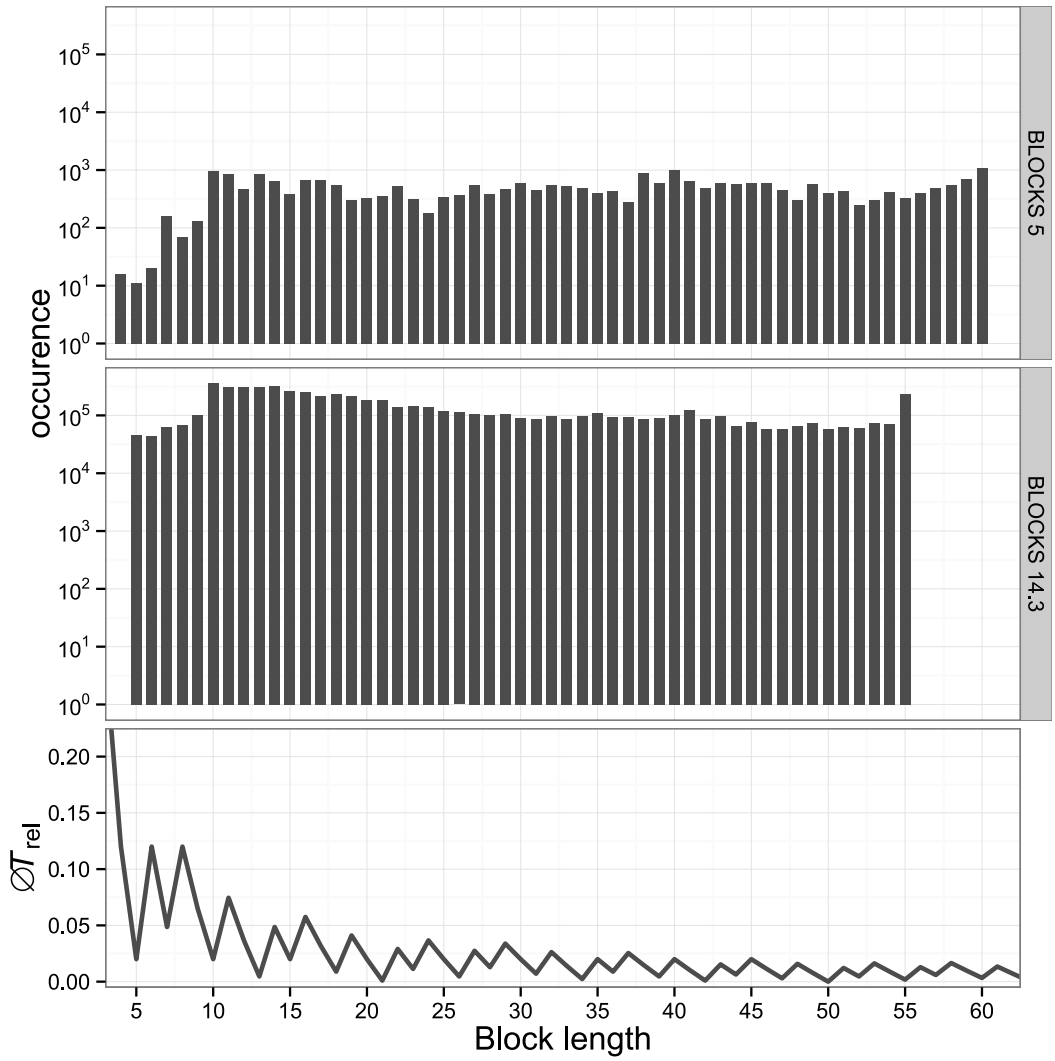


Figure 4.1: Theoretical error indicated by the relative threshold difference ΔT_{rel} for the similarity value $K = 62\%$. In the lowest panel, the difference ΔT_{rel} between the floating point threshold T and the truncated threshold $\widehat{T}$ is shown for increasing block lengths. The number of sequences found in the BLOCKS 5.0 and BLOCKS 14.3 databases for these block lengths are depicted in the panels above. The relative difference ΔT_{rel} is large for smaller blocks but vanishes with increasing block length. Note the systematic and therefore biased behavior of ΔT_{rel} as a function of the block length.

Compositional matrix differences

a) Analysis based on identical clustering thresholds

In order to analyze the impact of our CorBLOSUM correction (Listing 4.2) on the resulting matrix compositions in comparison to the original BLOSUM [Henikoff and Henikoff 1992a] and RBLOSUM [Styczynski et al. 2008] matrices, we first compared the entries of the CorBLOSUM matrices with their corresponding counterparts based on the same clustering threshold and originating BLOCKS version. Figure 4.2 shows the results of this evaluation in the form of the percentage of differing matrix entries. Notably, we omitted comparisons between matrices generated by clustering thresholds smaller than 15% since not all matrices in this range report valid log-odds scores for all pairs of amino acid types.

Independent of the originating database, one can see numerous entry changes between CorBLOSUM and (R)BLOSUM matrices indicating substantially different substitution matrices. In general, the number of differences between CorBLOSUM and BLOSUM matrices is larger than those between CorBLOSUM and RBLOSUM. This is not unexpected since the CorBLOSUM algorithm also incorporates RBLOSUM's error correction resulting in CorBLOSUM matrices being more similar to RBLOSUM matrices than BLOSUM matrices. Still, a large number of differences between CorBLOSUM and RBLOSUM matrices can be observed for all originating databases which highlights the substantial impact of using a floating point clustering threshold instead of an integer based threshold.

The largest number of changes can be observed for matrices created with smaller clustering thresholds where sequences are frequently clustered during the construction process even though they are relatively dissimilar. For example, the BLOCKS 5.0-based CorBLOSUM matrices constructed with clustering thresholds between 15% and 39% possess at least 50% different entries when compared to their $\text{BLOSUM}_{5.0}$ counterparts. Likewise, the comparison between $\text{CorBLOSUM}_{5.0}$ and $\text{RBLOSUM}_{5.0}$ in a similar clustering threshold range of $[15\%, 32\%]$ yields at least 34% different entries. In contrast, the number of differences between matrices based on larger thresholds tends to be much smaller. The comparison between CorBLOSUM matrices created with thresholds of $\geq 61\%$ based on BLOCKS 5.0 and their (R)BLOSUM counterparts, e.g., still yields $\sim 5\%$ to $\sim 18\%$ different entries.

Since the differences between CorBLOSUM matrices and their RBLOSUM and BLOSUM counterparts mainly depend on the number of inaccurate clustering decisions and not on the height of the chosen clustering threshold, the aforementioned observations have to be related to the sequence compositions of the tested BLOCKS databases. Apparently, the relative sequence similarity scores in the three assessed BLOCKS releases fall within specific scoring ranges. This results in more sequences being inaccurately clustered at certain clustering thresholds as indicated by the different "outliers" in Figure 4.2.

One interesting "outlier" is the comparison between $\text{CorBLOSUM59}_{5.0}$ and its (R)BLOSUM counterparts $\text{BLOSUM59}_{5.0}$ and $\text{CorBLOSUM59}_{5.0}$. The per-entry comparison of these matrices is shown in Figure 4.3 and reveals large differences not only in the number of differing entries but also in the magnitude of the log-odds scores. The upper triangle matrix depicts the per-entry differences between $\text{CorBLOSUM59}_{5.0}$ with $\text{RBLOSUM59}_{5.0}$, while the lower triangle shows the comparison between $\text{CorBLOSUM59}_{5.0}$ and $\text{BLOSUM59}_{5.0}$.

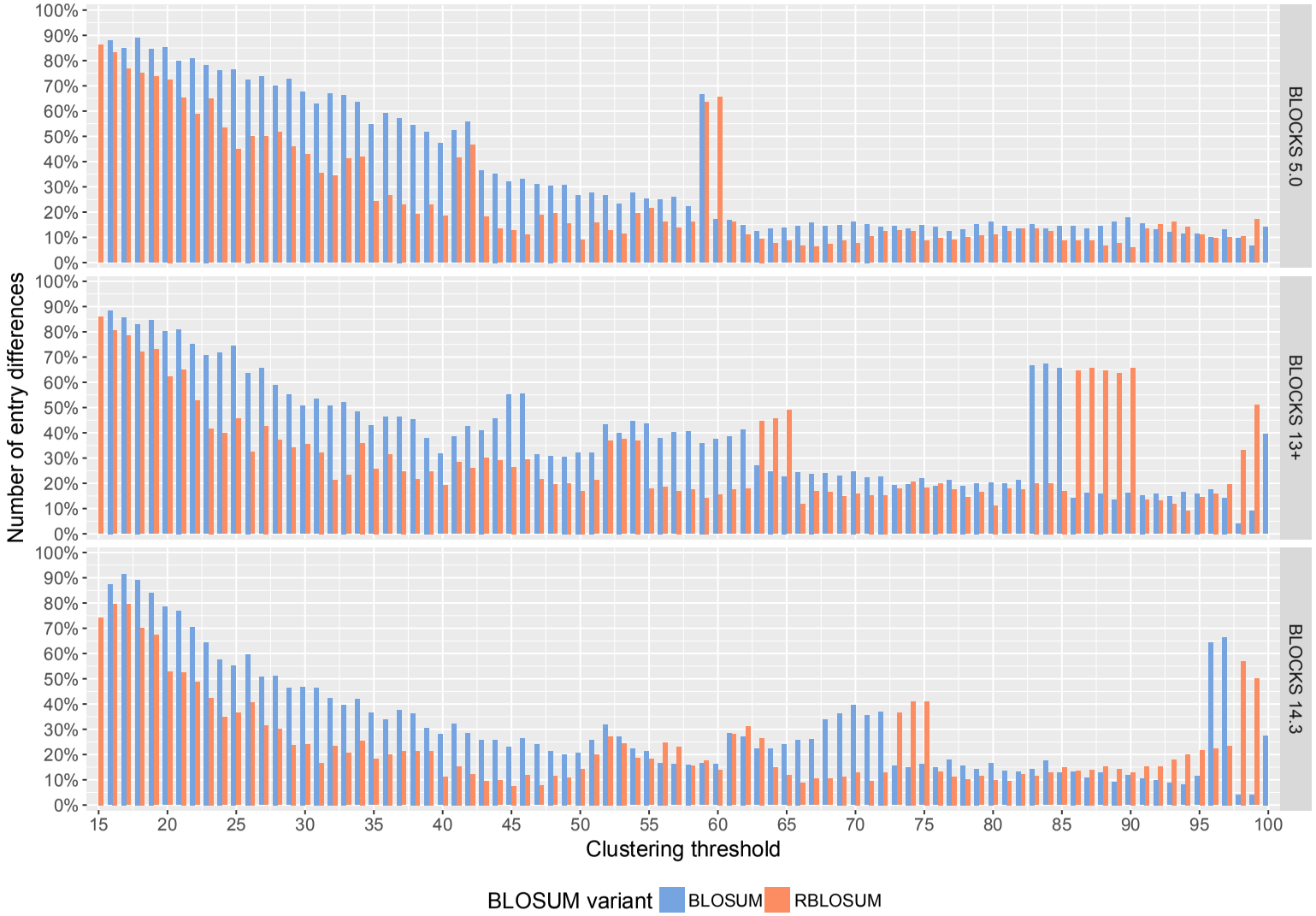


Figure 4.2: Compositional differences between CorBLOSUM and (R)BLOSUM matrices based on the same clustering value for different originating BLOCKS databases. Shown are the number of entry differences in percent when comparing a CorBLOSUM matrix with its (R)BLOSUM counterpart.

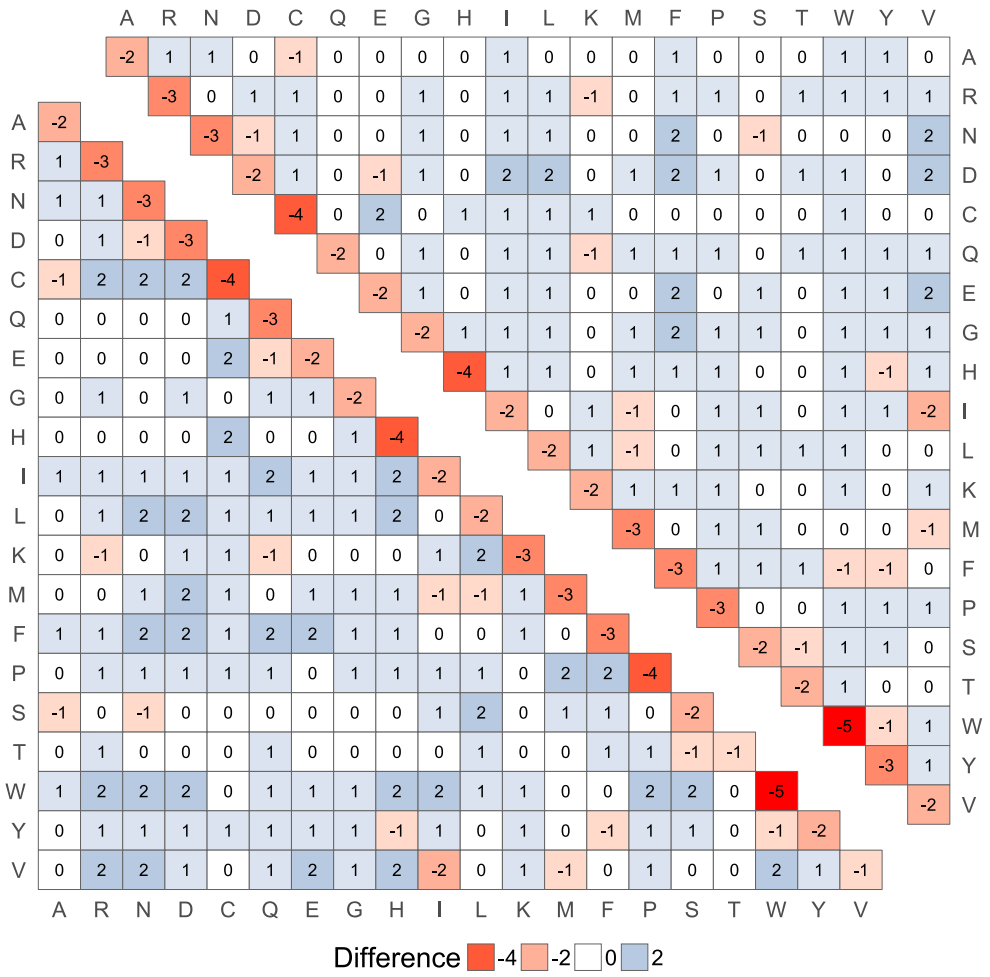


Figure 4.3: Per entry comparison of CorBLOSUM59$_{5.0}$ with BLOSUM59$_{5.0}$ (lower triangle) and RBLOSUM59$_{5.0}$ (upper triangle). Red values represent negative differences, i.e., entries being smaller in CorBLOSUM59$_{5.0}$ than its counterpart. Blue values show the opposite.

In both comparisons, the differences of the log-odds scores on the off-diagonal range between $[-2, 2]$. Interestingly, only $\sim 7.9\%$ of all log-odds scores on the off-diagonal of CorBLOSUM59$_{5.0}$ are smaller than their corresponding BLOSUM and RBLOSUM entries (red values). In contrast, $\sim 56.8\%$ (or $\sim 53.7\%$) of all CorBLOSUM59$_{5.0}$ off-diagonal values are larger than those in BLOSUM (or RBLOSUM) (blue values). The largest differences can be observed, however, on the diagonal with scoring differences between -1 and -5. The CorBLOSUM59$_{5.0}$ variant thus strongly favors more substitution events than its (R)BLOSUM counterparts BLOSUM59$_{5.0}$ and RBLOSUM59$_{5.0}$.

Since the BLOSUM62$_{5.0}$ matrix is arguably the most commonly used substitution matrices for homologous sequence search and the computation of multiple sequence alignments, we further compared the compositions of our CorBLOSUM62 matrices derived from the three different originating BLOCKS databases with their corresponding BLOSUM62 and RBLOSUM62 counterparts Figure 4.4.

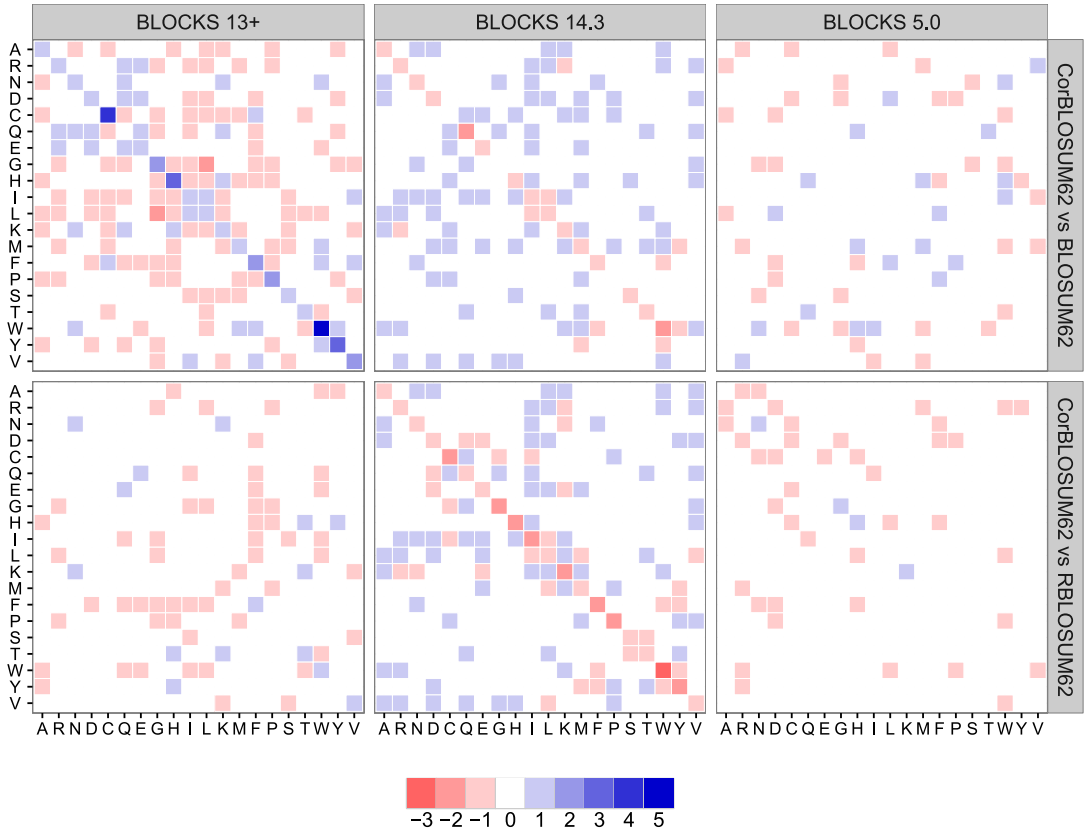


Figure 4.4: Per-entry comparisons between CorBLOSUM and (R)BLOSUM matrices derived from the three different BLOCKS databases using a clustering value of 62%. Red cells represent entries being smaller in CorBLOSUM62 than in its counterpart. Blue cells show those entries that are larger in CorBLOSUM62 when compared to the corresponding (R)BLOSUM counterpart.

Again, one can clearly see numerous changes between the matrices created by the three algorithms. For the BLOCKS 5.0 based substitution matrices, the log-odds scores differ in the range of −1 to 1 with most changes indicating smaller scores in the CorBLOSUM62$_{5.0}$ matrix when compared to (R)BLOSUM variants. In contrast, the BLOCKS 13+ and BLOCKS 14.3 based CorBLOSUM matrices differ more often from their counterparts and to a much greater extend with scoring differences from −3 to +5. Thus, changes in the matrices cannot exclusively be related to rounding issues. This provides further evidence that all three BLOSUM algorithms are substantially different.

b) Analysis based on relative entropy levels

While the compositional comparison of CorBLOSUM substitution matrices with BLOSUM and RBLOSUM matrices based on the same cluster threshold allows to reveal the differences introduced by the construction algorithms, the assessment of the performance difference between substitution matrices requires a different kind of evaluation. As per Altschul [1991], the performance of substitution matrices can only be compared in a fair way if they share similar relative entropies (H).

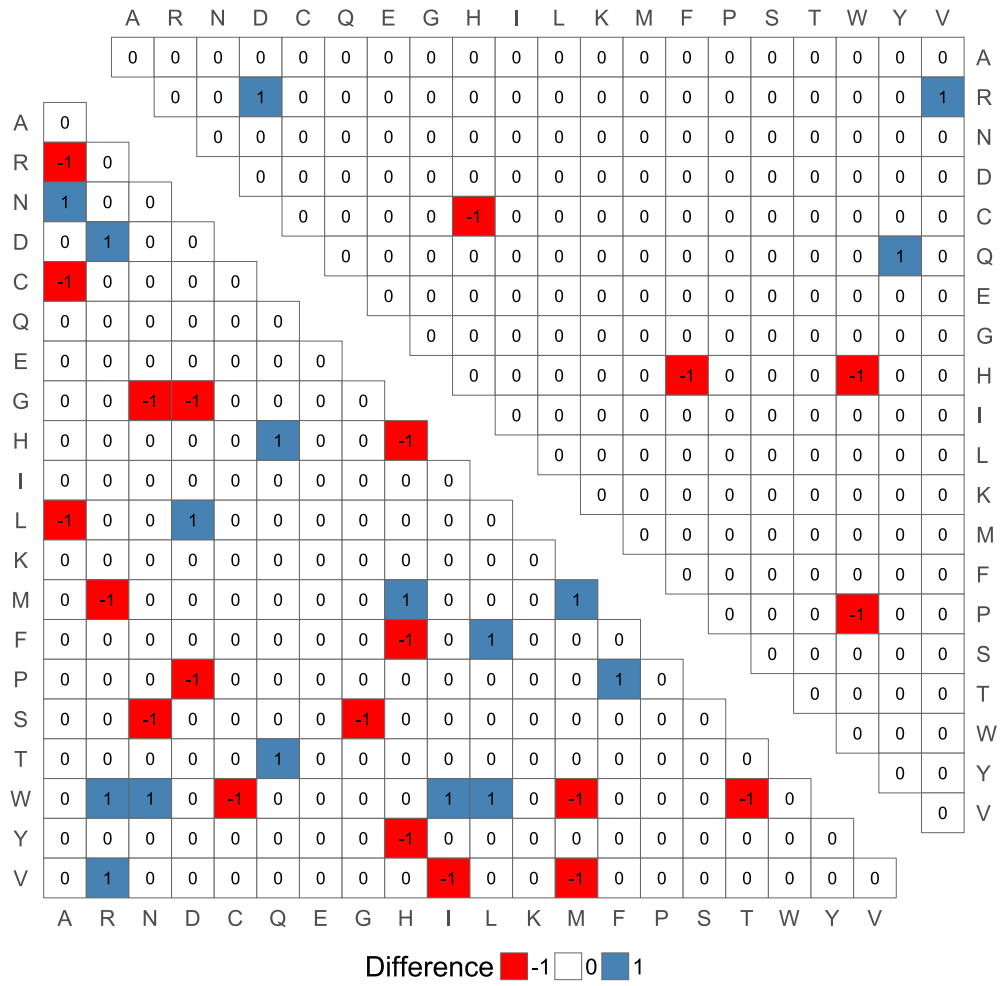


Figure 4.5: Per entry comparison of CorBLOSUM61$_{5.0}$ with the popular BLOSUM62$_{5.0}$ (lower triangle) and RBLOSUM64$_{5.0}$ (upper triangle). Red cells represent entries being smaller in CorBLOSUM61$_{5.0}$ than in its counterpart. Blue cells show those entries that are larger in CorBLOSUM61$_{5.0}$ than in the corresponding (R)BLOSUM counterpart.

For this reason and in concordance to the RBLOSUM study by Styczynski et al. [2008], we also compared the BLOCKS 5.0 based matrices CorBLOSUM61$_{5.0}$ ($H = 0.6939$), BLOSUM62$_{5.0}$ ($H = 0.6979$) and RBLOSUM64$_{5.0}$ ($H = 0.7003$) on a per-entry basis (Figure 4.5). Again, the differences between CorBLOSUM and its BLOSUM counterpart are shown at the lower triangle, while the upper triangle depicts the differences to the RBLOSUM-type matrix.

On one hand, a total of 31 matrix entries are different between the CorBLOSUM61$_{5.0}$ and BLOSUM62$_{5.0}$ (i.e., 14.8%), with 17 entries being reduced (Figure 4.5, lower triangle). On the other hand, only 7 entries differ between CorBLOSUM61$_{5.0}$ and RBLOSUM64$_{5.0}$, with three entries being larger in absolute value (Figure 4.5, upper triangle). The smaller number of differences between RBLOSUM64$_{5.0}$ and CorBLOSUM61$_{5.0}$ are not unexpected, as the RBLOSUM correction is also included in the CorBLOSUM algorithm. However, the number of differences between CorBLOSUM and RBLOSUM type matrices increases for other BLOCKS versions. The large differences between CorBLOSUM- and BLOSUM-type matrices

determined for BLOCKS 5.0 can also be observed for the other two BLOCKS releases. This comparison of matrices based on a similar entropy level further highlights that the three algorithms create substantially different substitution matrices.

Homology search performance

Method

In order to evaluate the different matrix variants BLOSUM, RBLOSUM and CorBLOSUM for the task of homologous sequence search, we employed the state-of-the-art methodology described in Section 4.3.1. Analogous to previous studies [Price et al. 2005, Styczynski et al. 2008, Song et al. 2015], we chose the ASTRAL database as the basis for our performance analysis.

As mentioned earlier, the performance study by Styczynski et al. [2008] was solely based on the ASTRAL40 1.69 release with less than 40% identical sequences. Instead, we tested the homology search performance of all generated substitution matrices on all available ASTRAL database releases (versions 1.55 to 2.06) in order to prevent potentially biased results caused by a specific database composition. For the same reason and inspired by Angermüller et al. [2012], we used three different sequence similarity thresholds (20%, 40% and 70%) for each release resulting in 51 separate benchmark datasets. In the following, we use the terms ASTRAL20, ASTRAL40, and ASTRAL70 to distinguish between these three similarity based subsets. Additionally, we use the terms SCOP- or SCOPe-based ASTRAL datasets to refer to ASTRAL versions 1.55 to 1.75 and 2.01 to 2.06, respectively. We would like to note that SCOP-based ASTRAL releases are entirely manually curated, while SCOPe releases are based on a semi-automated approach for the database generation.

This wide variety of databases allows to study the effect of improving sequence space coverage and different database compositions on matrix performance. However, performing this large number of benchmarks results in an immense computational effort. For this reason, we focused our performance analysis on a representative set of matrices derived from the three different BLOCKS database versions. Out of the original BLOSUM variants, we chose the clustering thresholds 50 and 62 since the $\text{BLOSUM50}_{5.0}$ and $\text{BLOSUM62}_{5.0}$ matrices are the two most widely used BLOSUM matrices. For example, these matrices are employed as default matrices in SSEARCH [Pearson 1991] and BLAST [Altschul et al. 1990]. The corresponding RBLOSUM and CorBLOSUM counterparts were selected according to their relative entropies in comparison to the chosen BLOSUM matrices in order to ensure a fair performance comparison [Altschul 1991]. The 18 matrices assessed in our study, their clustering values, relative entropies, and matrix scales based on unrounded log-odd scores are listed in Table 4.1.

Notably, the clustering thresholds of the $\text{RBLOSUM}_{5.0}$ and $\text{CorBLOSUM}_{5.0}$ matrices are closer to the one of the corresponding $\text{BLOSUM}_{5.0}$ matrices than to those based on BLOCKS 13+ and BLOCKS 14.3. This effect is induced by the different sequence compositions in the different BLOCKS releases. While the BLOCKS 5.0 release only provides 27,102 sequences for the matrix calculation, the BLOCKS 13+ provides 663,288 sequences and the even larger BLOCKS 14.3 database 6,739,916 sequences. Similarly, the composition of the database influences the relative matrix entropy. Whereas the entropy of the matrices which originate from BLOCKS 5.0 database is rather high, the distribution of substitution events (i.e., the

Matrix	Clust. Threshold	Rel. Entropy	Bit Units
$BLOSUM50_{5.0}$	50	0.4808	1/3
$RBLOSUM52_{5.0}$	52	0.4918	1/3
$CorBLOSUM49_{5.0}$	49	0.4849	1/3
$BLOSUM62_{5.0}$	62	0.6979	1/2
$RBLOSUM64_{5.0}$	64	0.7003	1/2
$CorBLOSUM61_{5.0}$	61	0.6939	1/2
$BLOSUM50_{13+}$	50	0.2430	1/4
$RBLOSUM59_{13+}$	59	0.2410	1/4
$CorBLOSUM57_{13+}$	57	0.2479	1/4
$BLOSUM62_{13+}$	62	0.3672	1/3
$RBLOSUM69_{13+}$	69	0.3601	1/3
$CorBLOSUM66_{13+}$	66	0.3653	1/3
$BLOSUM50_{14.3}$	50	0.1509	1/5
$RBLOSUM59_{14.3}$	59	0.1477	1/5
$CorBLOSUM57_{14.3}$	57	0.1515	1/5
$BLOSUM62_{14.3}$	62	0.2685	1/4
$RBLOSUM69_{14.3}$	69	0.2662	1/4
$CorBLOSUM67_{14.3}$	67	0.2636	1/4

Table 4.1: Overview of the matrices assessed in this study and their clustering values, relative entropies, and corresponding scale in bits per unit.

joint distribution) in the BLOCKS 13+ and BLOCKS 14.3 is closer to an independent event (i.e., the product of the marginals) and hence, the relative substitution matrix entropy is smaller.

In order to evaluate the performance of the different substitution matrices on the different ASTRAL databases, we conducted a homology search for each of the 51 ASTRAL databases against itself. Here, we used the Smith-Waterman alignment algorithm implemented in SSEARCH (version 36.3.6d) [Pearson 1991], as SSEARCH has been shown to possess higher accuracy than BLAST in assessing the performance of different substitution matrices [Henikoff and Henikoff 1992a, Green and Brenner 2002, Styczynski et al. 2008]. To address the potential bias from suboptimal gap penalty settings on the matrix performance, we varied the gap open penalty between 5 and 20 in steps of 1 and the gap extension penalty between 1 and 2. These penalties correspond to commonly used parameter settings in homology search tools (BLAST and SSEARCH) and previous performance studies such as [Price et al. 2005]. In total, we performed 29,376 entire database searches. This is, to our knowledge, the largest study of this kind performed till today.

For each combination of matrix, gap open and gap extension penalty, we obtained a list of homologs found for each sequence in the benchmarked ASTRAL release ordered by their *E*-value. The best performing gap parameter sets for each matrix on each of the tested ASTRAL databases are listed in Table B.1 till Table B.9. According to the state-of-the-art homology search benchmark methodology described in Section 4.3.1, we measured the

rate of correctly found homologs for a fixed errors-per-query limit (epq) of 0.01 using the quadratically normalized coverage measure $\mathcal{Q}_{quad}$. Similarly, the significance of the computed coverage values were examined by a Concerted Bayesian bootstrapping using 500 bootstrap rounds. All values were calculated using our `CoverageCalculator` program.

General matrix performance overview

In order to obtain a general overview, we counted how often a specific CorBLOSUM matrix performed equally or better than its corresponding BLOSUM counterpart. Considering all test scenarios, substitution matrices computed with the CorBLOSUM algorithm performed at least as good as their BLOSUM counterparts in $\sim$ 75% of the time. On SCOPe-based ASTRAL releases (versions 2.01 to 2.06) this percentage even increased to $\sim$ 86%.

Since we cannot directly compare the performance of substitution matrices derived from different BLOCKS versions due to their relative entropies, we compared the performance of each substitution matrix on all three similarity based ASTRAL subsets in an identical manner to the above described method. Cases where CorBLOSUM matrices performed at least as good as their corresponding BLOSUM variants are shown in Table 4.2. Here, the CorBLOSUM matrices performed better than the BLOSUM matrices, i.e., with one interesting exception, the original $\text{BLOSUM62}_{5.0}$ matrix. This matrix still performed better than its $\text{CorBLOSUM61}_{5.0}$ counterpart in most of the cases on the ASTRAL20 and ASTRAL40 subsets.

Although, the achieved coverage range differs widely between the ASTRAL20, ASTRAL40 and ASTRAL70 subsets, our results show a specific performance pattern within each identity subset regardless of the BLOCKS version and entropy level used for the computation of the matrices. For ASTRAL40 and ASTRAL70, the coverage increases drastically for ASTRAL versions based on SCOP (version $\leq$ 1.75) to those based on SCOPe (version $\geq$ 2.01). Interestingly, this trend cannot be observed for ASTRAL20.

BLOCKS version	**ASTRAL subset**	**BLOSUM50 entropy level**	**BLOSUM62 entropy level**
BLOCKS 5.0	ASTRAL20	70.59%	23.53%
	ASTRAL40	76.47%	23.53%
	ASTRAL70	100%	58.82%
BLOCKS 13+	ASTRAL20	94.12%	58.82%
	ASTRAL40	100%	76.47%
	ASTRAL70	100%	82.35%
BLOCKS 14.3	ASTRAL20	76.47%	76.47%
	ASTRAL40	76.47%	100%
	ASTRAL70	88.24%	70.59%

Table 4.2: Comparison of CorBLOSUM- with BLOSUM-type matrices. Shown is the relative frequency given in percent for which a CorBLOSUM matrix performed at least as good as its BLOSUM counterpart.

In the following sections, we discuss the matrix performances on each of the three different similarity based ASTRAL subsets in detail. The values reported there reflect the best matrix/-gap parameter combinations. For all test scenarios, we consider performance differences with Z-scores < 1.96 as insignificant and thus assume matrix performance to be almost equal. In cases where the coverage difference between a BLOSUM- and CorBLOSUM-type matrix is insignificant, an **O** is displayed above the bar. For the CorBLOSUM/RBLOSUM comparison, we highlight this with a small **X**. The underlying Z-scores for estimating the significance of these coverage differences are shown in Figure B.1, Figure B.2, and Figure B.3.

Matrix performance on ASTRAL40

For the ASTRAL40 subset results shown in Figure 4.6, a general performance trend can be observed for all assessed relative entropy levels. Starting from ASTRAL release 1.57 the performance increases steadily until ASTRAL 1.69, the database used by Styczynski et al. [2008] to measure the RBLOSUM performance. Here, a drastic drop in the coverages can be observed. From ASTRAL 1.71 the coverages continue to steadily increase with a very large increment upon the introduction of SCOPe at ASTRAL 2.01. The highest coverage over all entropy levels, BLOCKS versions and ASTRAL releases was obtained for CorBLOSUM$49_{5.0}$ on ASTRAL 2.06 with a coverage of 0.4389 at a gap open/extension penalty of 15/1.

For BLOCKS 5.0 derived substitution matrices at a matrix entropy level of ~ 0.7 bit, the original, inaccurate BLOSUM$62_{5.0}$ dominates the corrected variants on almost every ASTRAL release but the latest three. For these, CorBLOSUM$61_{5.0}$ and RBLOSUM$64_{5.0}$ performed at least as well as BLOSUM$62_{5.0}$ at a statistically significant level. Our results for the ASTRAL 1.69 database are in concordance with the results published in the RBLOSUM study [Styczynski et al. 2008] — i.e., the BLOSUM$62_{5.0}$ significantly outperforms the RBLOSUM$64_{5.0}$. Interestingly, the used BLOCKS version significantly influences this performance difference as RBLOSUM matrices derived from BLOCKS 13+ and BLOCKS 14.3 outperform their BLOSUM counterparts.

The CorBLOSUM$49_{5.0}$ showed higher coverages than the BLOSUM$50_{5.0}$ for all databases but the oldest ASTRAL and the oldest SCOPe derived ASTRAL databases 2.01 and 2.02. In general, BLOSUM$50_{5.0}$ entropy level matrices achieve higher coverages than those at the BLOSUM$62_{5.0}$ entropy level. This cannot be observed for BLOCKS 13+ and BLOCKS 14.3. For these, the CorBLOSUM57_{13+} and CorBLOSUM$67_{14.3}$ consistently outperformed their BLOSUM counterparts on all test databases. CorBLOSUM66_{13+} and CorBLOSUM$57_{14.3}$ achieved a coverage at least as high as the BLOSUM in $\sim 76\%$ of the tested scenarios. For all SCOPe derived ASTRAL datasets CorBLOSUM substitution matrices outperformed their BLOSUM counterpart.

Overall, the comparison between CorBLOSUM- and RBLOSUM-type matrices showed mixed results. Notably, CorBLOSUM matrices derived from BLOCKS 13+ and BLOCKS 14.3 achieved higher coverages than RBLOSUM matrices in $\sim 83\%$ of the analyzed SCOPe-based datasets.

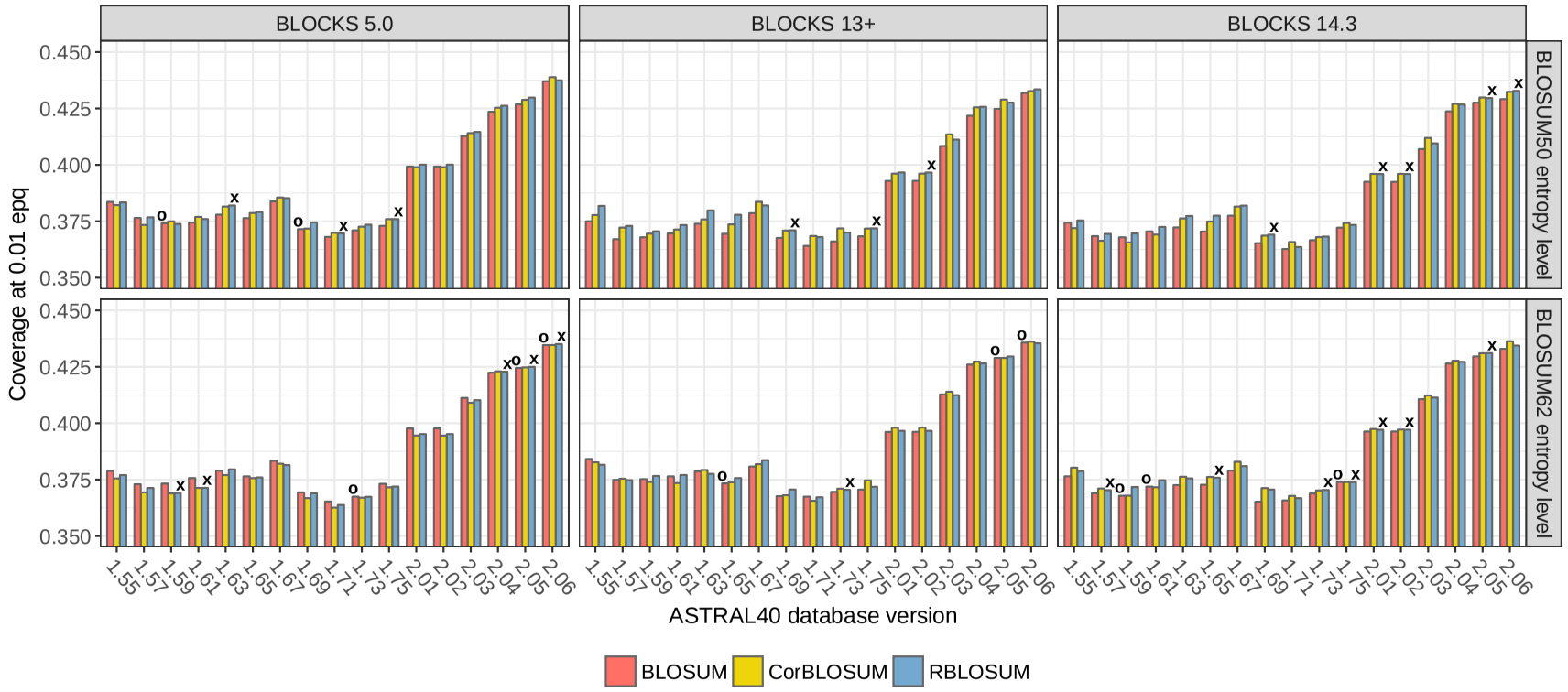


Figure 4.6: Progression of the maximum achieved coverage of CorBLOSUM-, RBLOSUM- and BLOSUM-type matrices for all ASTRAL40 test databases. The upper row shows the results for the BLOSUM50 entropy level and the lower row depicts the BLOSUM62 entropy level. Insignificant coverage differences between CorBLOSUM and BLOSUM are indicated by an **O** and between CorBLOSUM and RBLOSUM by a small an **X** above the bars. The corresponding gap parameter settings are listed in Table B.4, Table B.5, and Table B.6. Notably, the coverage dramatically increases for all tested substitution matrices with the introduction of the semi-automatic database generation of SCOPe. On the BLOSUM50 entropy level, CorBLOSUM-type matrices performed at least as good as their BLOSUM and RBLOSUM counterparts in $\sim$ 84% and $\sim$ 49% of all tested scenarios, respectively. On the BLOSUM62 entropy level, CorBLOSUM matrices showed equal or better performance than BLOSUM/RBLOSUM in $\sim$ 67%/$\sim$ 60% of all analyzed ASTRAL40 scenarios.

Matrix performance on ASTRAL70

The coverage results for the different versions of the ASTRAL70 subset are shown in Figure 4.7. Overall, the matrix performances on this subset showed the highest coverage scores in our homology search performance benchmark with values of up to 0.5445. This coverage value was achieved on the latest ASTRAL70 release (version 2.06) by our CorBLOSUM66$_{13+}$ in combination with a gap opening/extension penalty of 12/1.

When analyzing the coverage values in relation to the different ASTRAL versions, one can see a similar general performance trend to the results reported for the ASTRAL40 subset. Starting from the initial ASTRAL release 1.55, the coverage scores decrease slowly until release 1.59 and then increase steadily until ASTRAL 1.69. Here, the same drastic drop in the coverage scores determined for ASTRAL40 can be observed for the ASTRAL70 subset. Likewise, one can also discover the huge increment observed for ASTRAL40 at the ASTRAL version 2.01 — the introduction of SCOPe.

In contrast to ASTRAL40 where the original BLOCKS 5.0-based BLOSUM62$_{5.0}$ dominated their corrected variants on almost all ASTRAL releases but the latest three, the corresponding results on ASTRAL70 mostly do not show significant performance differences. Also, CorBLOSUM49$_{5.0}$ always achieved as least as high coverage values than BLOSUM50$_{5.0}$ and thus outperforms the inaccurate original BLOSUM variant on average. In comparison to the BLOCKS 5.0-based RBLOSUM matrices, again, one can see mixed results similar to the results obtained for ASTRAL40.

Our results obtained for the BLOCKS 13+-based matrices demonstrate that CorBLOSUM matrices perform at least as good or better than BLOSUM matrices in nearly all cases. Especially on SCOPe-based datasets, CorBLOSUM matrices outperformed their BLOSUM counterparts significantly. In comparison to the BLOCKS 13+-based RBLOSUM matrices, however, CorBLOSUM matrices deliver similar coverage scores with negligible performance differences.

The comparison of the BLOCKS 14.3-based matrices with a relative entropy level comparable to BLOSUM50$_{14.3}$ showed that CorBLOSUM57$_{14.3}$ outperformed BLOSUM50$_{14.3}$ in most cases — especially on SCOPe-based databases. However, the comparison to RBLOSUM59$_{14.3}$ revealed a slight performance advantage of RBLOSUM59$_{14.3}$ over CorBLOSUM57$_{14.3}$. For the BLOSUM62 entropy level, the different matrix types showed overall mixed results.

In general, the comparison for the BLOSUM50 entropy level shows that CorBLOSUM-type matrices perform at least as good as their original BLOSUM counterparts in $\sim$ 94% of all test cases. In comparison to the RBLOSUM variants on the same entropy level, CorBLOSUM matrices still showed similar or better performance in $\sim$ 51% of the tested scenarios. On the BLOSUM62 entropy level, CorBLOSUM matrices were able to perform as least as good as their BLOSUM and RBLOSUM counterparts in $\sim$ 75% and $\sim$ 59% of the tested databases, respectively.

Matrix performance on ASTRAL20

Figure 4.7 depicts the coverage results obtained for the different versions of the ASTRAL20 subset. The ASTRAL20 coverage results are the smallest scores measured in our benchmark with a value of only 0.1634 achieved for the ASTRAL20 1.55 database by RBLOSUM69$_{13+}$ and gap opening/extension penalties of 12/1. This is not unexpected since a maximum

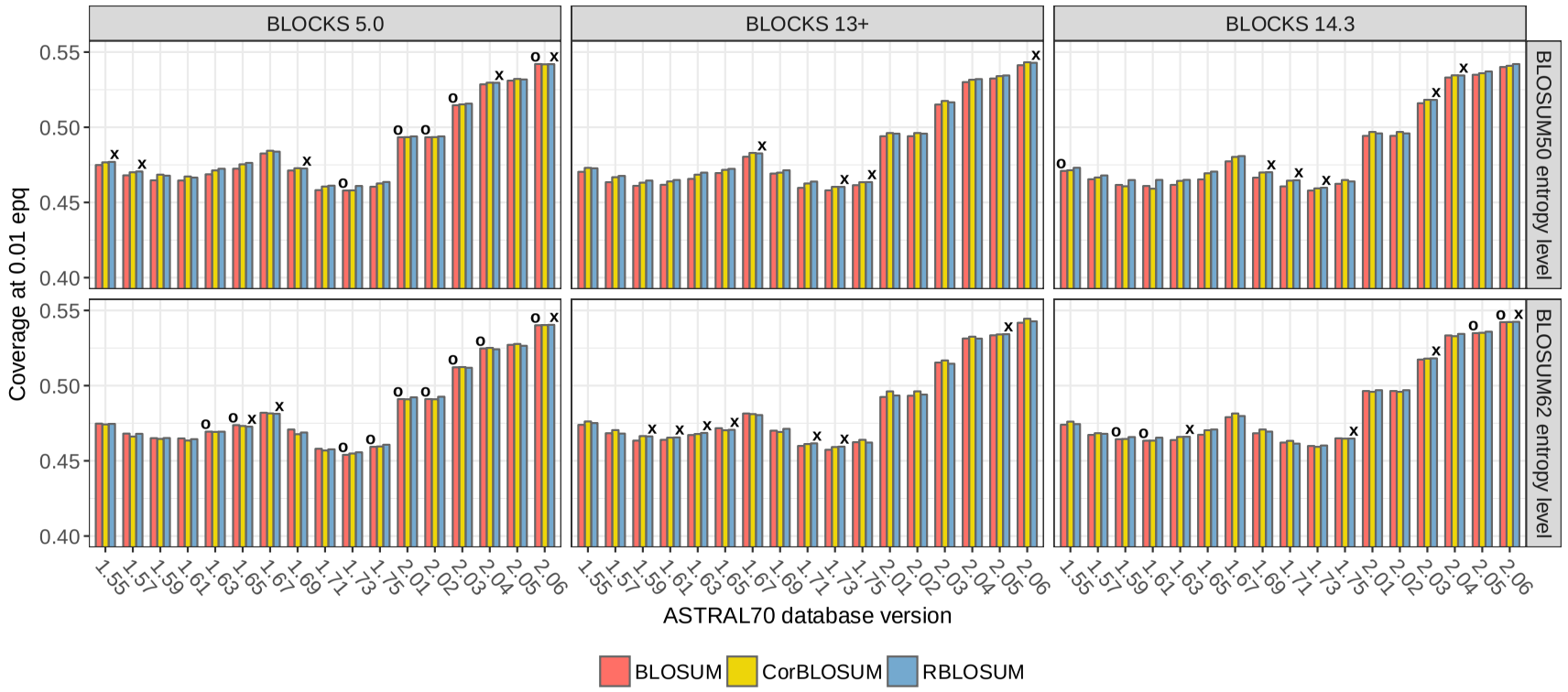


Figure 4.7: Progression of the maximum achieved coverage of CorBLOSUM-, RBLOSUM- and BLOSUM-type matrices for all ASTRAL70 test databases. The upper row shows the results for the BLOSUM50 entropy level and the lower row shows the results for the BLOSUM62 entropy level. An insignificant coverage difference between CorBLOSUM and BLOSUM is indicated by an **O** and between CorBLOSUM and RBLOSUM by an **X**. The corresponding gap parameter settings are listed in Table B.7, Table B.8, and Table B.9. Similar to the ASTRAL40 test scenarios, a drastic increase in coverage can be observed for SCOPe-based ASTRAL databases. On the BLOSUM50 entropy level, CorBLOSUM-type matrices performed at least as good as their BLOSUM counterparts in $\sim$ 94% of all test cases and in $\sim$ 51% when compared with RBLOSUM. On the BLOSUM62 entropy level, CorBLOSUM matrices showed equal or better performance than BLOSUM/RBLOSUM in $\sim$ 75%/$\sim$ 59% of all analyzed ASTRAL70 scenarios.

sequence similarity of only 20% represents the scenario of identifying distantly related homologs which is generally a more difficult task than searching homologs in similar sequence datasets.

Interestingly, the highest coverage scores for this ASTRAL subset were mostly achieved on the oldest ASTRAL release 1.55. This is quite different to the results obtained for the other ASTRAL subsets where the highest coverages were always reported for the latest ASTRAL release 2.06. Likewise, the general performance trend observed for all matrices on the ASTRAL40 and ASTRAL70 differs to a greater extend from the trend visible in the ASTRAL20 subset. Here, the coverage values steadily decrease with few exceptions starting from ASTRAL version 1.55 until version 2.04. The drastic performance increase upon the introduction of SCOPe (version 2.01) observed on ASTRAL40 and ASTRAL70, can first be seen on ASTRAL20 for the SCOPe-based ASTRAL version 2.04.

When investigating the performance of the BLOCKS 5.0-based matrices on ASTRAL20, one can see a similar behavior for $\text{BLOSUM62}_{5.0}$ compared to the results obtained for ASTRAL40. The original BLOCKS 5.0-based $\text{BLOSUM62}_{5.0}$ dominated their corrected variants on almost all ASTRAL releases but the latest three. For the BLOSUM50 entropy level, we obtained mixed results for all matrix types on older ASTRAL releases. However, on newer ASTRAL20 databases we observe a significant performance advantage of $\text{CorBLOSUM49}_{5.0}$ over $\text{BLOSUM50}_{5.0}$. With the exception of the latest ASTRAL20 version 2.06, $\text{CorBLOSUM49}_{5.0}$ also significantly outperforms $\text{RBLOSUM64}_{5.0}$.

Our analysis of the BLOCKS 13+-based matrices revealed that the CorBLOSUM_{13+} matrices outperformed the BLOSUM_{13+} matrices in nearly all cases. Especially on SCOPe-based datasets, CorBLOSUM matrices always superseded their BLOSUM counterparts significantly. Interestingly, BLOCKS 13+-based RBLOSUM matrices perform better than CorBLOSUM matrices on older ASTRAL20 releases but are significantly outperformed on ASTRAL20 versions ≥ 2.03.

On the $\text{BLOSUM50}_{14.3}$ entropy level, our $\text{CorBLOSUM57}_{14.3}$ matrix significantly superseded their BLOSUM and RBLOSUM counterpart for ASTRAL20 versions ≥ 1.69. Overall, the results obtained for the $\text{BLOSUM62}_{14.3}$ entropy level show mixed results. An interesting exception are the coverage scores obtained for the recent ASTRAL20 releases 2.04 and 2.05. Here, the $\text{BLOSUM62}_{14.3}$ showed the best performance of all matrices. For the latest ASTRAL20 version 2.06, however, no performance difference between our $\text{CorBLOSUM67}_{14.3}$ and $\text{BLOSUM62}_{14.3}$ can be seen.

In summary, using our tested CorBLOSUM variants for homologous sequences search on the three latest ASTRAL20 subsets (2.04, 2.05, and 2.06) resulted in significantly more correctly found homologs compared to the tested (R)BLOSUM matrices. There are only three exceptions to this finding. $\text{RBLOSUM57}_{14.3}$ superseded $\text{CorBLOSUM57}_{14.3}$ on ASTRAL20 2.06. On the ASTRAL20 subset versions 2.04 and 2.05, $\text{BLOSUM62}_{14.3}$ performed better than $\text{CorBLOSUM57}_{14.3}$ and $\text{RBLOSUM59}_{14.3}$.

4.4.4 Conclusion

We presented an additional error correction to the BLOSUM code. The matrices created by our CorBLOSUM algorithm are substantially different from (R)BLOSUM matrices and outperformed the original BLOSUM matrices in $\sim 75\%$ of all 51 test scenarios. On up-to-date

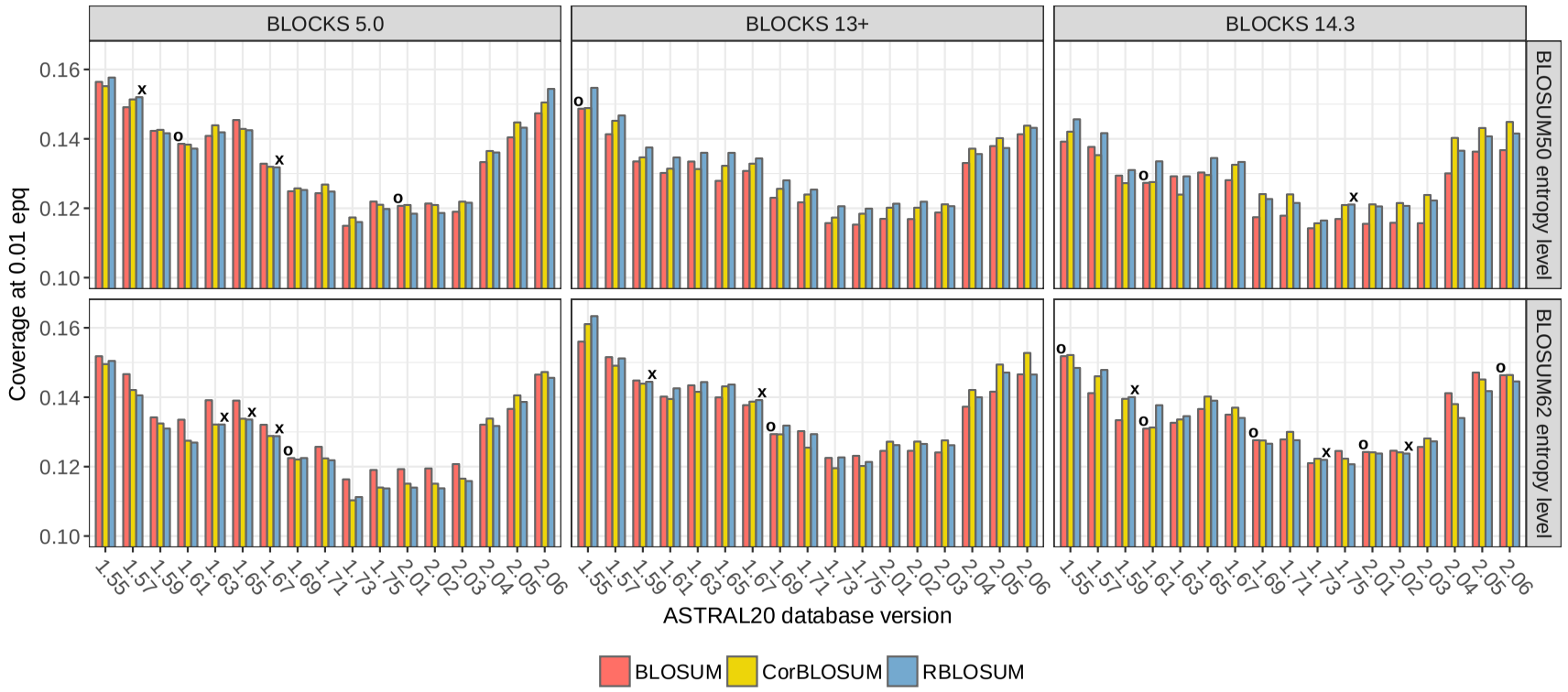


Figure 4.8: Progression of the maximum achieved coverage of CorBLOSUM-, RBLOSUM- and BLOSUM-type matrices for all ASTRAL20 test databases. The upper row shows the results for the BLOSUM50 entropy level and the lower row depicts those for the BLOSUM62 entropy level. An insignificant coverage difference between CorBLOSUM and BLOSUM is indicated by an **O** and between CorBLOSUM and RBLOSUM by an **X**. The corresponding gap parameter settings are listed in Table B.1, Table B.2, and Table B.3. The BLOSUM62 entropy level substitution matrices derived from BLOCKS_{13+} and $\text{BLOCKS}_{14.3}$ consistently achieved higher coverages than those on the BLOSUM50 entropy level. On the BLOSUM50 entropy level, CorBLOSUM-type matrices performed at least as good as their BLOSUM and RBLOSUM counterparts in $\sim$ 80% and $\sim$ 53% of all tested scenarios, respectively. On the BLOSUM62 entropy level, CorBLOSUM matrices showed equal or better performance than BLOSUM/RBLOSUM in $\sim$ 49%/$\sim$ 70% of all analyzed ASTRAL20 scenarios.

SCOPe-based ASTRAL releases, the current gold standard for homology search performance assessment, the CorBLOSUM matrices outperformed their BLOSUM counterparts in ∼ 86% of the cases. On these databases, the CorBLOSUM matrices also achieved the highest reported coverages for all three ASTRAL similarity subsets when compared with their BLOSUM counterparts.

The aim of this study was not to assess optimal parameters for homologous sequence search, such as the best matrix/gap-parameter combination. Nevertheless, this is an interesting question that should be addressed in the future, especially since our study showed that the relative entropy of substitution matrices is not necessarily an indicator for matrix performance.

Our results for the BLOSUM62$_{5.0}$ vs. RBLOSUM64$_{5.0}$ setup concur with the previous findings by Styczynski et al. [2008]. There, the test covered only a very specific scenario (ASTRAL40 1.69) in which the RBLOSUM64$_{5.0}$ was outperformed by the BLOSUM62$_{5.0}$. These previous results would have been quite different if at that time other available BLOCKS and ASTRAL databases had been used. RBLOSUM matrices tested in this study performed equally or better than their BLOSUM counterparts in most of the cases. Our study showed that for the RBLOSUM/CorBLOSUM comparison no consistent trend can be observed for older ASTRAL releases prior to 2.01, as RBLOSUM and CorBLOSUM matrices are superior in ∼ 50% of these cases. However, on databases with increased sequence and structure space coverage – as provided by SCOPe-based ASTRAL versions – CorBLOSUM-type matrices achieved higher coverages than the RBLOSUM matrices in ∼ 74% of the tests.

Furthermore, our study revealed two contradicting effects: on the one hand, matrices with very similar entropies show a statistically significant difference in performance. On the other hand, we also showed that matrices with very different entropies and matrix scales can achieve similar coverages. The latter effect is apparently enhanced by increasing sequence similarity within superfamilies and the database itself. This raises an interesting question for further research on the influence of changes in database composition on its respective searchability.

We conclude that the CorBLOSUM algorithm fixes errors of the original BLOSUM implementation and that the resulting matrices perform better for homologous sequence search. Hence, we encourage the usage of CorBLOSUM matrices for this specific task.

4.5 PFASUM substitution matrices

4.5.1 Introduction

Most of the widely used substitution matrices to date are based on quite old, filtered, and small datasets (Section 4.2). As thoroughly discussed in the previous section, the popular BLOSUM matrix is even substantially biased through programming errors in their originating source code. These issues may severely limit the sensitivity of these substitution matrices for the construction of sequence alignments compared to matrices derived from a larger and more diverse sequence space [Price et al. 2005, Hess et al. 2016a].

For these reasons, we developed a novel type of substitution matrix based on structural alignments. Our **Pf**am **su**bstitution **m**atrix (PFASUM) series is derived from the manually curated Pfam seed alignments (version 29.0) [Finn et al. 2016] using a novel algorithm. Thus,

our PFASUM matrices rely on state-of-the-art expert ground truth data that covers a much larger and diverse sequence space than conventional substitution matrices. As of release 29.0, there are 47.3 billion amino acid pairings available in 16,295 MSAs. Additionally, our PFASUM algorithm takes all information in the MSAs into account instead of omitting parts containing gaps or ambiguous amino acids to avoid the loss of important information. These features enable PFASUM matrices to significantly outperform commonly used substitution matrices, especially when dealing with sequences of low similarity.

In the following, we will first describe the construction methodology of our PFASUM matrices in detail. Subsequently, we present a thorough performance evaluation and discussion of PFASUM's capabilities for homologous sequence search and MSA construction in comparison to frequently used substitution matrices. The results of this evaluation provide evidence for our claims that PFASUM matrices deliver superior performance over state-of-the-art matrices for the tasks of homologous sequence search and MSA construction.

4.5.2 Algorithm

Database selection

As outlined in the introduction, commonly used substitution matrices are derived from quite old and incomplete (filtered) datasets. It is also known that larger and more diverse sequence datasets can produce significantly better performing substitution matrices [Price et al. 2005, Hess et al. 2016a]. Hence, we chose the Pfam seed alignment dataset [Finn et al. 2016] as the basis for our PFASUM substitution matrices. This dataset consists of numerous MSAs (16,295 MSAs in Pfam release 29.0 [Finn et al. 2016]) that cover the currently known sequence space. Each MSA contains a set of representative sequences for a specific group of proteins such as a protein family or domain. This allows our algorithm to capture substitution events between closely related sequences. Combining all MSAs, and thus different groups of sequences, into a single matrix enables us to apply derived substitution events on protein sequences with distant relationships. Furthermore, all Pfam MSAs are manually curated by experts and thus represent ground truth structural alignments.

PFASUM algorithm

As described in the introduction of this chapter (Section 4.1), substitution matrices usually represent substitution rates in the form of rounded log-odds scores derived from aligned and filtered sequence data. For two different amino acids α_i and α_j the unrounded score S_{α_i,α_j} corresponds to $S_{\alpha_i,\alpha_j} = \log_2 p(\alpha_i,\alpha_j) - \log_2\big(p(\alpha_i)p(\alpha_j)\big)$. The term $p(\alpha_i,\alpha_j)$ denotes the substitution frequencies for α_i and α_j which are derived by counting all $\alpha_i\alpha_j$ pairings $n(\alpha_i,\alpha_j)$ and relating these to all counted pairs, i.e., $N = \sum_{\alpha_i,\alpha_j} n(\alpha_i,\alpha_j)$. The terms $p(\alpha_i)$ and $p(\alpha_j)$ represent the marginals for observing amino acid α_i and α_j, respectively. These can directly be derived by summing over the probability of conservation $p(\alpha_i,\alpha_i)$ and all substitution events ($\sum_{j\neq i} p(\alpha_i,\alpha_j)$). Often, the resulting real-numbered log-odds score S_{α_i,α_j} is rounded to the next integer value.

Our PFASUM algorithm and the corresponding matrices also follow this basic principle. We process each MSA in the Pfam seed dataset separately, accumulate the counted substitution frequencies in a single matrix, and subsequently transform these to the final rounded log-

odds scores. In order to process unfiltered Pfam seed alignments and to handle special cases such as oversampling issues and ambiguous amino acids, it is, however, necessary to introduce additional steps. The PFASUM algorithm thus consists of five major components:

- A **sequence clustering** step to prevent oversampling based on the method proposed by Henikoff and Henikoff [1992].
- A **sequence similarity measure** to determine cluster memberships that copes with different sequence lengths, jointly proposed by *Frank Keul* and *Martin Heß*.
- A **strategy for handling gaps** frequently found in Pfam seed MSAs, jointly proposed by *Frank Keul* and *Martin Heß*.
- A **group size normalization** step to further mitigate potential bias from oversampling caused by different sequence group sizes proposed by *Martin Heß*.
- A method for computing log-odds scores for **ambiguous amino acids** proposed by *Frank Keul*.

Sequence clustering

Counting amino acid substitutions in a set of highly redundant sequences may result in potential bias from oversampling. To mitigate this problem, Henikoff and Henikoff [1992] use a clustering algorithm for the BLOSUM-t matrix calculation to group sequences of equal length λ depending on their relative similarity Φ and a preset clustering threshold t. Instead of counting the observed amino acid substitutions $n(\alpha_i, \alpha_j)$ for each sequence pair A and B fully, each substitution is counted as $n(\alpha_i, \alpha_j)/(|c_x| * |c_y|)$. Thereby, c_x corresponds to the cluster that contains sequence A and c_y to the cluster containing sequence B. The cardinalities $|c_x|$ and $|c_y|$ represent the corresponding cluster sizes, i.e., the number of sequences within the clusters. If both sequences A and B belong to the same cluster, i.e., $c_x = c_y$, all substitutions between A and B are ignored in counting pairs. In other words, a single cluster is considered as a single sequence in counting pairs.

First, the clustering algorithm calculates the similarity $\Phi(\mathbf{A}, \mathbf{B})$ between two sequences A and B. The similarity $\Phi(\mathbf{A}, \mathbf{B})$ between two sequences A and B is measured by counting the number of aligned positions that share the same amino acid type, normalized by the length λ of both sequences. The similarity value is then compared to the preset clustering threshold t in order to decide whether the sequences should be grouped or not. For example, if sequences A and B are $\Phi(\mathbf{A}, \mathbf{B}) = 73.5\%$ identical and the clustering threshold is set to $t = 62$, the sequences are grouped within a cluster. Additional sequences C are assigned to this cluster if at least one sequence X exists inside the cluster that is at least t % similar to C, i.e., $\Phi(\mathbf{C}, \mathbf{X}) \geq t$.

The PFASUM algorithm incorporates this method to mitigate oversampling problems. Analogous to the BLOSUM matrices, the number suffix in a matrix' name (e.g., PFASUM43) indicates the chosen clustering threshold used for the construction of the matrix. However, we had to adapt the similarity measure to cope with the aligned sequences found in the Pfam seed dataset as explained in the following section.

Sequence similarity

Two aligned sequences A and B from a Pfam seed alignment can directly be compared as both can be considered as correctly aligned. A gap symbol found in these alignments is denoted as γ. We define $\mathbf{A}$ as a vector of amino acid and gap symbols with the length L, i.e., $\mathbf{A} = (\alpha_1, \ldots, \alpha_L)$. The number of amino acid symbols in $\mathbf{A}$, i.e., without the gap symbol γ, is denoted as λ_A. Similarly, $\mathbf{B}$ is defined as $(\beta_1, \ldots, \beta_L)$ and λ_B represents the number of amino acids in $\mathbf{B}$. The unnormalized similarity ϕ between $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\phi(\mathbf{A}, \mathbf{B}) = \sum_{l=1}^{L} \delta(\alpha_l, \beta_l)[1 - \delta(\alpha_l, \gamma)][1 - \delta(\beta_l, \gamma)] \tag{4.4}$$

Hereby, $\delta(x, y)$ is the Kronecker delta which equals one if $x = y$, and zero otherwise. In other words, we omit all pairings that contain at least one gap symbol and count only pairs with identical one letter codes. The fractional similarity value Φ is computed by normalizing $\phi(\mathbf{A}, \mathbf{B})$ with the length of the shorter sequence, i.e., $\min(\lambda_A, \lambda_B)$.

Group size normalization

Generally, Pfam alignments represent groups of related protein regions. For our purpose, these groups of sequences are collections of related proteins. Within the broad term “group” we encapsulate protein families, domains, and similar organizational structures. Substitution matrices derived from Pfam seed alignments aim at capturing the average evolutionary behavior, while the number of sequences within each group can vary widely. To avoid over-representing groups with high sequence counts, the PFASUM algorithm derives group-specific pair frequency counts $p_k(\alpha_i, \alpha_j)$ for each sequence group k. These $p_k(\alpha_i, \alpha_j)$ are obtained by normalizing the pair counts $n_k\left(\alpha_i, \alpha_j\right)$ found in a group k with the number of sequences s_k in this group. Pair frequencies $p(\alpha_i, \alpha_j)$ for the entire database are then obtained by summing over all normalized $p_k(\alpha_i, \alpha_j)$.

Gaps

The Pfam seed dataset contains complete MSAs and thus gaps occur frequently to compensate for different sequence lengths induced by deletions and insertions of amino acids. This is contrary to the data basis of most conventional substitution matrices, e.g., the BLOCKS database used for the BLOSUM construction [Henikoff and Henikoff 1991]. These datasets are usually filtered by omitting alignment parts containing gaps. Rather than neglecting MSA columns with at least one gap, the PFASUM algorithm simply neglects gap/amino acid (as well as gap/gap) pairings in counting substitution frequencies. Hence, the PFASUM algorithm considers all found amino acid pairings even in gap-rich columns with few amino acids. Since these regions are also manually curated and thus can be considered as reliably aligned, this allows us to extract unique information about substitution events even within insertion/deletion regions (indels).

Ambiguous amino acids

Ambiguous amino acid characters – such as B, Z, J, and X – occur rarely in most sequence databases, especially in older databases. This consequently results in very low frequencies for pairs that involve ambiguous amino acids so that the computed relative pair frequencies often vanish. Hence, most substitution matrix algorithms fully ignore observed ambiguous

amino acids when counting pair frequencies. Instead, matrix entries for these characters are subsequently generated from averaging the pair frequencies of the canonic amino acids, following the translation scheme shown in Table 4.3.

The number of ambiguous amino acids can be, however, larger in modern sequence databases such as Pfam and thus can have a greater influence on the observed substitution frequencies. To correctly account for this, PFASUM fully processes ambiguous amino acids in counting substitution frequencies. As ambiguous amino acids encode at least two canonic amino acids, it is necessary to redefine amino acids as a set Θ_x of symbols with x representing the one letter code of the amino acid. Canonic amino acids are thus represented as sets with a cardinality of one, e.g., $\Theta_A = \{A\}$. Ambiguous amino acids are defined as sets containing their encoded canonic acids, e.g., $\Theta_B = \{N, D\}$.

The PFASUM algorithm equally distributes pair counts $\Theta_x\Theta_y$ of any found ambiguous amino acid among their canonic amino acids, again following Table 4.3. Also, we count the observed $\Theta_x\Theta_y$ directly. For example, each found amino acid substitution $\Theta_A\Theta_B$ in group k is counted as 0.5 $\Theta_A\Theta_N$, 0.5 $\Theta_A\Theta_D$, and 1.0 $\Theta_A\Theta_B$.

Afterwards, the final pair frequency counts for all acid-to-acid combinations x, y are obtained using the following formula which accounts for counting $\Theta_x\Theta_y$ more than once if an ambiguous amino acid is involved:

$$\mu_k\left(\Theta_x, \Theta_y\right) = \frac{\left[\sum_{\alpha_i \in \Theta_x} \sum_{\alpha_j \in \Theta_y} n_k\left(\alpha_i, \alpha_j\right)\right] - n_k\left(\Theta_x, \Theta_y\right)}{|\Theta_x||\Theta_y|}$$
$$p_k\left(\Theta_x, \Theta_y\right) = \frac{\mu_k\left(\Theta_x, \Theta_y\right) + n_k\left(\Theta_x, \Theta_y\right)}{N_k} \tag{4.5}$$

In p_k, we add a correction term $\mu_k(\Theta_x, \Theta_y)$ to the number of observed amino acid pairings $n_k\left(\Theta_x, \Theta_y\right)$. This term equates to zero for pairings of canonic amino acids. For pairs including ambiguous amino acids, μ copes with adding these pairings to both canonic amino acids pair counts as well as ambiguous acid pair counts. This ensures that each observed amino acid pairing is only counted once. The normalization factor N_k corresponds to the total number of observed amino acid pairings in a group k.

Using the example from above, a single observed AB substitution would result in frequency counts $n_k(\Theta_A, \Theta_N) = n_k(\Theta_A, \Theta_D) = 0.5$ and $n_k(A, B) = 1$. The resulting relative frequencies using Equation 4.5 then yield $p_k(\Theta_A, \Theta_N) = p_k(\Theta_A, \Theta_D) = 0.5$ and $p_k(\Theta_A, \Theta_B) = 1$.

Ambiguous amino acid	B	Z	J	X
Canonic amino acid	N, D	E, Q	I, L	all

Table 4.3: Ambiguous amino acids and their designated canonic amino acids shown as one letter codes.

4.5.3 Evaluation - Homology search performance

One of the most common tasks employing substitution matrices is the identification of homologous sequences. We thus evaluated the capabilities of PFASUM matrices for this particular task in three different scenarios. First, we analyzed the performance differences between PFASUM matrices generated by different clustering values to get a general overview of PFASUM's homology search performance. Second, we compared the performance of PFASUM matrices with commonly used substitution matrices (e.g., BLOSUM62) and R/CorBLOSUM matrices based on similar relative entropy levels. This scenario represents the state-of-the-art approach for comparing substitution matrices as per Altschul [1991] on the basis of their general compositional properties (and to some extend the underlying algorithm). Third, we investigated homology search performance in a more user centered way by analyzing which of the tested matrices performed best on different databases, regardless of their relative entropies. The following sections describe the different matrix test sets and databases used for this evaluation as well as the employed methodology in detail.

Tested substitution matrices

We calculated PFASUM matrices using integer clustering values ranging from 0 to 100. Depending on the input data, too small clustering thresholds can lead to clustering results only containing a single large "super-cluster". Since amino acid substitutions are only counted between different clusters and not within the clusters, this can result in matrices that do not report substitution rates for all possible amino acid substitutions. Hence, we omitted all PFASUM matrices with clustering thresholds < 10. We denote the finally obtained matrix set in the following as *PFASUM Search Matrices* (Table 4.4).

In order to evaluate the homology search performance of *PFASUM Search Matrices* against state-of-the-art substitution matrices, we focus on a set of widely used substitution matrices denoted as *Standard Search Matrices* (Table 4.4). This set contains various BLOSUM, MD, Optima, PAM, and VTML matrices which are used, e.g., as default parameter in popular homology search tools such as SSEARCH/FASTA [Pearson 1991] and BLAST [Altschul et al. 1990].

In addition, we computed a set of (R/Cor)BLOSUM, PAM, and VTML matrices that possess similar relative entropies to the PFASUM31, PFASUM43, and PFASUM60 matrices, the three best performing *PFASUM Search Matrices* on the tested databases as shown later in the first part of this evaluation. This set is denoted as *PFASUM-comparable Search Matrices* and allows us to compare the performance of PFASUM matrices with our previously proposed CorBLOSUM and state-of-the-art matrices based on similar relative entropy (Table 4.4).

Databases

To investigate the performance differences between *PFASUM Search Matrices* and *Standard Search Matrices* or *PFASUM-comparable Search Matrices*, we conducted homology search experiments on basis of the most recent ASTRAL 2.06 [Brenner et al. 2000, Chandonia et al. 2004] datasets, a subset of sequences of the SCOP/SCOPe database [Murzin et al. 1995, Fox et al. 2014]. As outlined in Section 4.3.1, the ASTRAL database, and especially its ASTRAL40 subset containing sequences with a maximum similarity of 40%, has been commonly suggested as the gold standard for homology search performance evaluation

[Brenner et al. 1998, Green and Brenner 2002, Price et al. 2005, Styczynski et al. 2008]. In order to evaluate substitution matrix performance for different application purposes, we differentiated between three distinct test scenarios and thus ASTRAL database subsets.

While the ASTRAL40 subset represents the state-of-the-art homology search benchmark, the ASTRAL70 dataset emulates database searches against very similar and closely related sequences. We also conducted a performance evaluation based on the ASTRAL20 dataset. This simulates homology searches of novel proteins with unknown structural features and few known homologs. Effectively, these three datasets allow us to evaluate the performance of substitution matrices for different evolutionary distances.

Search methods

To obtain the most accurate results for the evaluation of homology search performance we employed the SSEARCH [Pearson 1991] algorithm of FASTA (Version: 36.3.8d). SSEARCH was reported previously to possess higher accuracy than BLAST [Henikoff and Henikoff 1992a, Green and Brenner 2002, Styczynski et al. 2008]. In order to avoid a potential bias introduced through inaccurate gap parameter settings, we varied the gap opening and gap extension penalties from -5 to -20 and -1 to -3, respectively. For each gap parameter and substitution matrix combination, we generated a list of found potential homologous relations when searching all sequences of an ASTRAL dataset to the entire database. These relations were ordered based on their E-values, i.e., the probability of obtaining a hit for an unrelated sequence with equal length by pure chance.

Performance evaluation

Similar to our CorBLOSUM performance benchmark (Section 4.4.3), we evaluated the homology search performance of substitution matrices using the state-of-the-art approach by using the quadratically normalized coverage measure $\mathcal{Q}_{quad}$ described in Section 4.3.1. For a list of homologous search results ordered by their E-values, this measure represents the fraction of the correctly found, true positive superfamily relations which remain after

Test set	Algorithm	Matrix numbers
PFASUM Search Matrices	PFASUM	[11,100]
Standard Search Matrices	BLOSUM	50, 62, 80
	MD	10, 20, 40
	Optima	5
	PAM	120, 250
	VTML	10, 20, 40, 80, 120, 160, 200
PFASUM-comparable Search Matrices	BLOSUM	37, 43, 51
	RBLOSUM	37, 43, 52
	CorBLOSUM	35, 41, 50
	PAM	203, 258, 316
	VTML	182, 226, 270

Table 4.4: The matrix test sets assessed in the homology search performance evaluation.

cutting the list in order to restrict the number of false positives to a certain amount. We set this threshold to 0.01 errors per query (epq) in concordance to other studies [Price et al. 2005, Hess et al. 2016a]. This effectively restricts the number of false positives found within 100 queries to a single false positive.

The significance of the differences between the coverage scores obtained for different matrices were also examined using the state-of-the-art approach by employing Concerted Bayesian bootstrapping and Z-score statistics (Section 4.3.1). Concerted Bayesian bootstrapping allowed us to evaluate the effect of changes in the database composition on the resulting coverage scores. Through variation of the prior distribution of the sequences in the database, we obtained 500 different coverage scores for each list of search results computed with a particular matrix and gap parameters. The significance of the performance difference between two of these benchmark results was then examined by computing a Z-score based on the mean and variance of these coverage distributions and the number of performed bootstrap steps, i.e., 500 in our case.

Overview - PFASUM homology search performance

In the first part of our evaluation, we investigated the homology search performance of all *PFASUM Search Matrices* on each of the three ASTRAL subsets. The highest coverage values obtained for our *PFASUM Search Matrices* with clustering values ≥ 15 in combination with optimal gap penalties are shown in Figure 4.9. The full list of obtained coverage scores for all *PFASUM Search Matrices* and the corresponding optimal gap penalties can be found in Table C.5, Table C.6, and Table C.7.

Similar to the results of our CorBLOSUM performance study presented in Section 4.4.3, the largest coverage values were reported on the ASTRAL70 subset with scores of up to ~ 0.5508, closely followed by the results obtained for ASTRAL40 with coverages of up to ~ 0.4448. Likewise, the overall smallest coverage scores with a maximum of ~ 0.1706 were obtained for the ASTRAL20 subset. Again, this is not unexpected because identifying true homologs in dissimilar sequence datasets is more difficult than finding homologous sequences in closely related sequence sets.

When analyzing the results for ASTRAL20 in detail, the overall largest coverage scores were obtained for *PFASUM Search Matrices* with clustering values between 34 and 64. In this range, the highest coverage score of $\Omega \approx 0.1706$ was achieved by the PFASUM60 matrix and a gap open/extension penalty of -16/-1, closely followed by PFASUM48 ($\Omega \approx 0.1701$) and PFASUM47 ($\Omega = 0.1695$) for gap parameters of -15/-1.

The results for the ASTRAL40 subset show a similar performance trend. On this subset, the largest coverage scores were achieved by *PFASUM Search Matrices* with a clustering value range of $\sim [30, 60]$. Here, PFASUM43 in combination with gap open/extension penalty settings of -13/-1 produced the highest coverage score with $\Omega \approx 0.44483$. The second and third highest coverage scores were produced by PFASUM45 ($\Omega \approx 0.44476$) and PFASUM41 ($\Omega \approx 0.44471$) using gap penalties of -13/-1 and -14/-2, respectively.

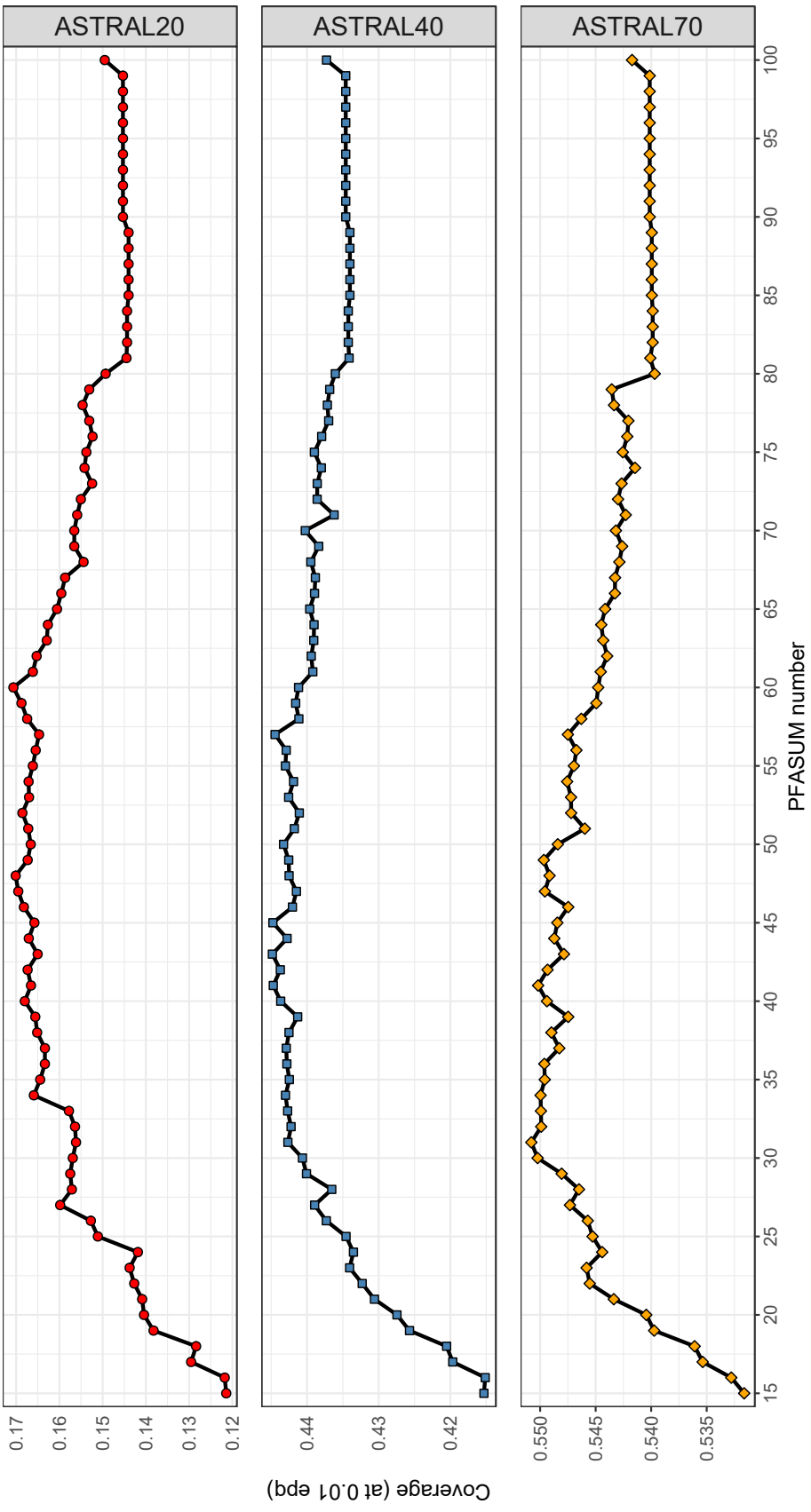


Figure 4.9: Overview of the homology search performance of PFASUM-type matrices on the three tested ASTRAL 2.06 subsets. Each data point represents the highest coverage score obtained for a particular PFASUM matrix and corresponding optimal gap parameters (Table C.5, Table C.6, and Table C.7). The lines indicate the progression of the achieved coverage values in dependency of increasing PFASUM clustering values on the three ASTRAL subsets.

On ASTRAL70, the largest coverage scores were obtained for matrices with a clustering value range of $\sim [29, 50]$ with PFASUM31 being the best performing matrix. This matrix achieved the highest coverage of ~ 0.5508 in combination with a gap open penalty of -13 and an extension penalty of -2. The performance differences to the other *PFASUM Search Matrices* in the clustering value range of $\sim [29, 50]$ is, however, rather small.

In general, *PFASUM Search Matrices* with small clustering values (≤ 24) delivered the worst homology search performance across all tested ASTRAL subsets. Another interesting performance pattern can be observed for *PFASUM Search Matrices* with clustering values ≥ 80. All matrices in this range delivered nearly the same performance with the exception of PFASUM100 which shows a slight performance increase over its predecessors. Between these ranges ($24 < \Omega < 80$), *PFASUM Search Matrices* show similar performance trends on all subsets. This indicates that PFASUM matrices can be useful for the detection of closely related sequences as well as distant homologs.

Comparison based on relative entropy

In order to properly assess the performance of substitution matrices, Altschul [1991] suggested comparing matrices with similar relative entropy H, the information divergence between independent and observed evolutionary relations. For this reason, we compared the homology search performance of our PFASUM matrices with the capabilities of state-of-the-art substitution matrices of comparable entropy levels in two different scenarios.

In the first scenario, we compare the three best performing PFASUM matrices on the tested ASTRAL subsets with their PAM, VTML, and (R/Cor)BLOSUM counterparts denoted as *PFASUM-comparable Search Matrices*. This enables us to judge the performance of the best PFASUM matrices in comparison to our own CorBLOSUM matrices and those generated with the most popular substitution matrix algorithms. The second scenario is focused on the comparison of *Standard Search Matrices*, i.e., those matrices widely used for homologous sequence search, with their PFASUM counterparts.

a) Best performing PFASUM matrices vs. (R/Cor)BLOSUM, PAM, and VTML

In this first scenario, we compare the performance of the best performing *PFASUM Search Matrices* on the three ASTRAL subsets (PFASUM31, PFASUM43, and PFASUM60) with their PAM, VTML, and (R/Cor)BLOSUM counterparts of comparable relative entropies. The matrices assessed in this scenario are listed with their relative entropies in Table 4.5.

BLOSUM, RBLOSUM, and CorBLOSUM matrices were constructed using the respective variations of the original BLOSUM code [Henikoff and Henikoff 1992b] described in Section 4.4.2. The VTML matrices were generated using the original VT/VTML scripts[2] provided by Tobias Mueller. The PAM matrices were computed using the C program `pam`[3] by E. Michael Gertz and Stephen F. Altschul. Since this program neither reports relative entropy values nor computes log-odds scores for ambiguous amino acids, we modified the source code accordingly. Notably, the log-odds scores of ambiguous amino acids computed with our implementation do not always match their corresponding entries found in existing PAM matrices, e.g., those

[2] The perl scripts used for the generation of the VTML matrices were obtained from `https://owww.molgen.mpg.de/~muelle_t/vt_scores`, last accessed 05.07.2016.

[3] The `pam` source code was obtained from `ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/pam.tar.gz`, last accessed 20.06.2017.

PFASUM31 $H \approx 0.230$		**PFASUM43** $H \approx 0.335$		**PFASUM60** $H \approx 0.494$	
Matrix	**H**	**Matrix**	**H**	**Matrix**	**H**
BLOSUM37	0.231	BLOSUM43	0.337	BLOSUM51	0.485
CorBLOSUM35	0.234	CorBLOSUM41	0.337	CorBLOSUM50	0.488
RBLOSUM37	0.231	RBLOSUM43	0.333	RBLOSUM52	0.492
PAM316	0.229	PAM258	0.335	PAM203	0.495
VTML270	0.229	VTML226	0.334	VTML182	0.496

Table 4.5: Table of (R/Cor)BLOSUM, VTML, and PAM matrices with comparable relative entropies to the best performing *PFASUM Search Matrices* on the three ASTRAL subsets PFASUM31, PFASUM43, PFASUM60.

obtained from the NCBI FTP server[4]. We relate these slight score offsets to numerical differences between intermediate results computed with our implementation and the unknown original source code. Since our implementation of the relative entropy computation does not take ambiguous amino acids into account and these acid types rarely occur in the tested ASTRAL subsets, we still consider our benchmark results for these matrices as valid.

Figure 4.10 shows the highest achieved coverage scores at 0.01 errors per query (epq) on the three ASTRAL subsets for the PFASUM31, PFASUM43, and PFASUM60 matrices and *PFASUM-comparable Search Matrices* each using optimal gap penalties(Table C.8). The significance of the results was estimated with Z-score statistics based on the mean and variance of coverage distributions generated by Concerted Bayesian bootstrapping using 500 bootstrap steps (Table C.8).

Our results demonstrate that PFASUM31 and PFASUM43 always significantly outperform their (R/Cor)BLOSUM, VTML, and PAM counterparts on all three tested ASTRAL subsets and thus are better options for homologous sequence search than their counterparts. PFASUM60 also significantly supersedes its counterparts on ASTRAL20 but is in turn outperformed by VTML182 on the ASTRAL40 and ASTRAL70 subsets. On these datasets, however, PFASUM60 is also outperformed by PFASUM31 and PFASUM43. This indicates that PFASUM60 is generally not the best *PFASUM Search Matrices* for homologous sequence search on databases with high sequence similarity.

b) Standard Search Matrices vs. PFASUM counterparts

For the second evaluation scenario, we identified *PFASUM Search Matrices* with similar relative entropies to the *Standard Search Matrices* set. Since the entropy values of *PFASUM Search Matrices* (Table C.4) range between 0.0668 bits (PFASUM11) and 0.7319 bits (PFASUM100) and some *Standard Search Matrices* possess to drastically different or unknown relative entropies, we could not directly compare all *Standard Search Matrices* to our *PFASUM Search Matrices*. The full list of *Standard Search Matrices* and their comparable PFASUM counterparts based on similarity entropy levels assessed in this scenario is shown in Table 4.6.

[4]NCBI FTP server `ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/`, last accessed 20.06.2017.

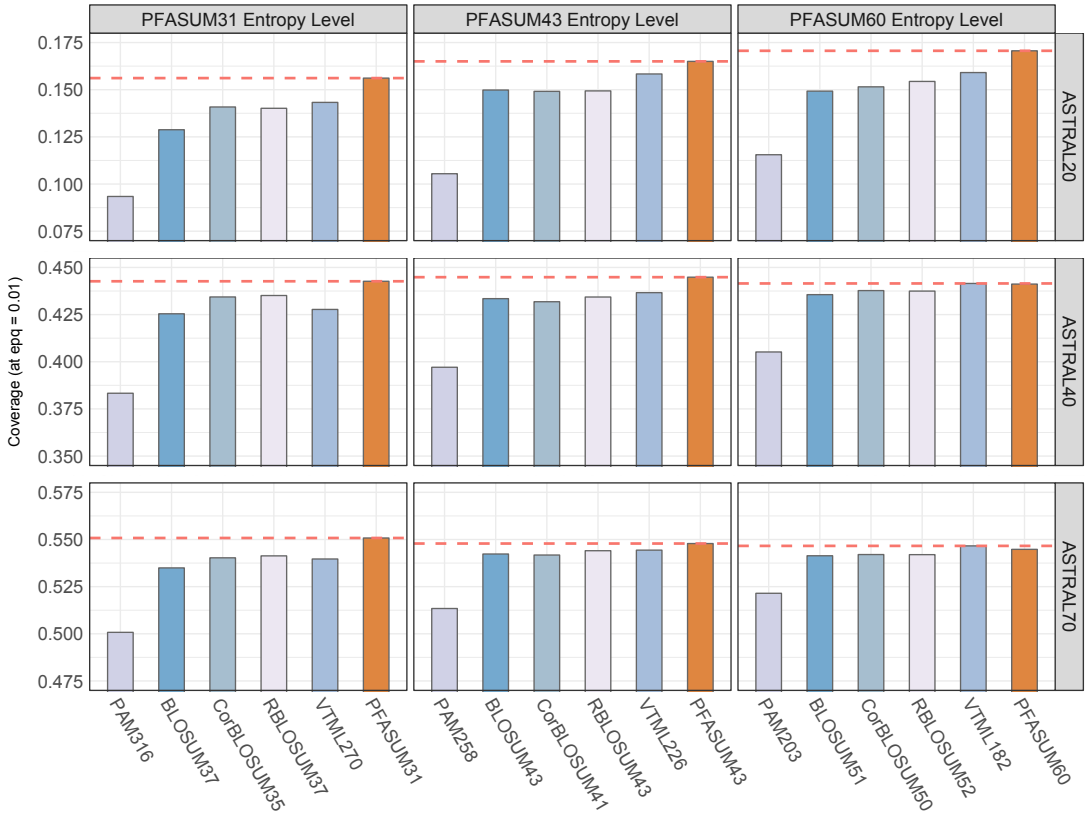


Figure 4.10: Performance comparison of the best performing *PFASUM Search Matrices* on the three ASTRAL subsets (PFASUM31, PFASUM43, and PFASUM60) with their PAM, VTML, and (R/Cor)BLOSUM counterparts with comparable relative entropies (*PFASUM-comparable Search Matrices*). The highest achieved coverage at 0.01 errors per query for the best gap opening and extension penalty combination is shown. The dashed red line indicates the maximum coverage value obtained for each comparison. Notably, the shown range of the coverage is reduced to emphasize the differences between the matrices.

For all *Standard Search Matrices* matrices with comparable entropy levels to *PFASUM Search Matrices* matrices, we compared their performance on the three ASTRAL datasets for varying gap penalty settings. Figure 4.11 shows the highest achieved coverage scores at 0.01 errors per query (epq) for *Standard Search Matrices* and their comparable PFASUM counterparts each using individual best performing gap penalties (Table C.9). Again, the significance of the results was estimated with Z-score statistics and Concerted Bayesian bootstrapping (Table C.9).

The obtained coverage results and Z-Scores show that *PFASUM Search Matrices* always perform at least as good or significantly better than their comparable *Standard Search Matrices* with one single exception: The VTML160 matrix performs slightly better on the ASTRAL70 dataset. This performance difference can be related to the different matrix compositions. While the diagonal entries of VTML160 and its counterpart PFASUM67 are very similar, there are numerous differences of up to four log-odds scores in PFASUM67 when comparing the off-diagonal entries. PFASUM67 thus favors more substitution events

than VTML160 which may be useful when searching for remote homologs. For datasets containing similar sequences such as ASTRAL70, however, this can result in more false positive relationships identified.

On a global level, the performance advantage of *PFASUM Search Matrices* over the tested *Standard Search Matrices* grows with decreasing sequence similarity in the test databases. While the performance differences on ASTRAL70 and ASTRAL40 are only marginal but still significant, the coverage differences for ASTRAL20 are much greater. This indicates that *PFASUM Search Matrices* are especially useful for detecting remote homologs.

Entropy-independent search performance comparison

Henikoff and Henikoff [1993] showed that substitution matrices of a given matrix family perform best around a relative entropy H of ~ 0.7 bit. We chose to re-evaluate this hypothesis on the basis of the Pfam seed database. Similar to Hess et al. [2016], our results show that the best performing substitution matrices – including *PFASUM Search Matrices* – possess relative entropies well below the suggested 0.7 bit.

As shown in Figure 4.12, the best performance on the ASTRAL20 dataset was achieved by PFASUM60 (Table C.3) with a relative entropy of $H = 0.4941$ bit. The best performing matrices on the datasets ASTRAL40 and ASTRAL70 are PFASUM43 ($H = 0.3354$ bit, Table C.2) and PFASUM31 ($H = 0.2297$ bit, Table C.1).

Standard Search Matrix	Entropy (bit)	PFASUM Search Matrix	Entropy (bit)
BLOSUM50	0.4808	PFASUM59	0.4849
BLOSUM62	0.6979	PFASUM78	0.6931
BLOSUM80	0.9868	n/a	n/a
MD10	n/a	n/a	n/a
MD20	n/a	n/a	n/a
MD40	n/a	n/a	n/a
Optima5	n/a	n/a	n/a
PAM120	0.9790	n/a	n/a
PAM250	0.3540	PFASUM45	0.3529
VTML10	3.4680	n/a	n/a
VTML20	2.9125	n/a	n/a
VTML40	2.2675	n/a	n/a
VTML80	1.4279	n/a	n/a
VTML160	0.5625	PFASUM67	0.5649
VTML200	0.4121	PFASUM51	0.4084

Table 4.6: Table of *Standard Search Matrices* with their relative entropy as listed in FASTA (Version: 36.3.8d, found in `upam.h`). Comparable PFASUM substitution matrices are listed with their respective entropy. Entries with n/a refer to comparisons where no PFASUM matrix with comparable relative entropy level could be found or the relative entropy of the *Standard Search Matrices* is unknown.

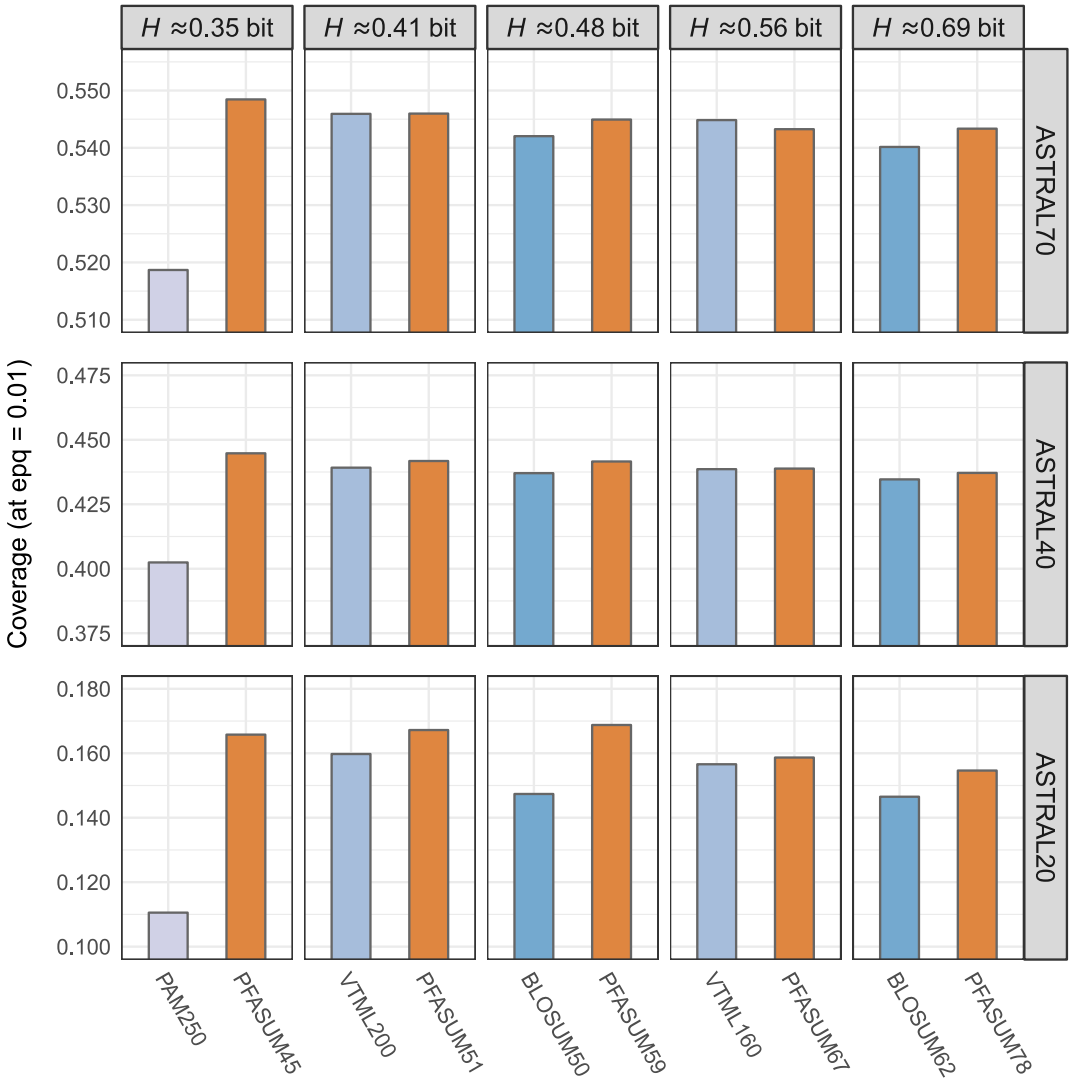


Figure 4.11: Performance comparison of *Standard Search Matrices* with *PFASUM Search Matrices* of similar entropies on all three ASTRAL datasets. Shown is the highest achieved coverage at an 0.01 errors per query for any gap opening and extension penalty combination. The best-performing gap parameter for each matrix and database combination as well as the corresponding Z-scores can be found in Table C.9. Notably, the shown range of the coverage is reduced to emphasize the differences between the matrices.

We also find that PFASUM matrices with higher matrix number tend to perform better on sequence data with lower sequence similarity than matrices with lower matrix number (Figure 4.9). Whereas PFASUM60 shows the best performance on the ASTRAL20 dataset, the best performing PFASUM matrices for sequences with relatively high sequence similarity can be found at low clustering thresholds with PFASUM31 which outperforms all the others on the ASTRAL70 dataset.

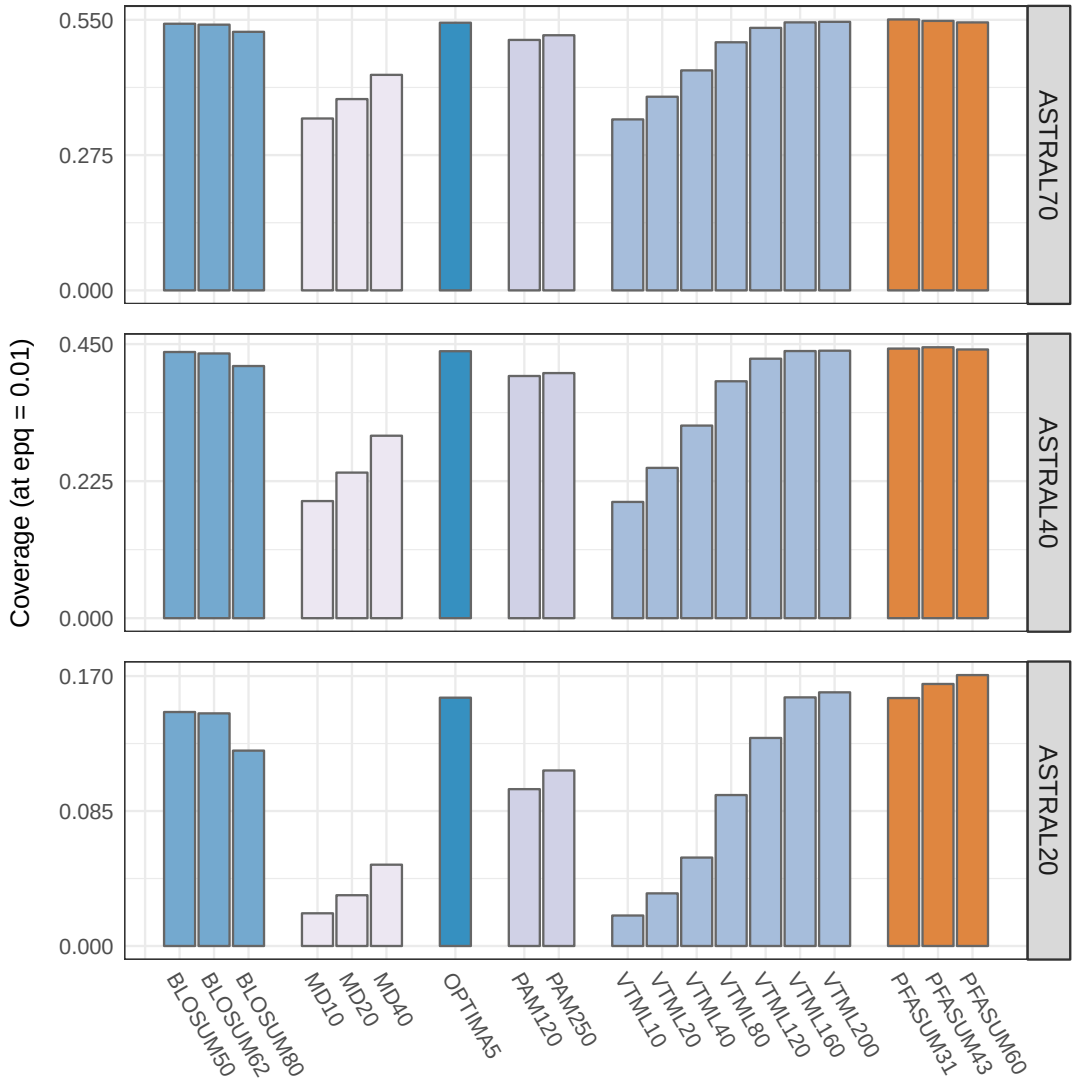


Figure 4.12: Comparison of the performance of all *Standard Search Matrices* with the novel *PFASUM Search Matrices* on three different ASTRAL datasets. Shown are the highest achieved coverage scores at 0.01 errors per query for any gap opening and extension penalty combination (Table C.10). With the exception of two performance differences, all shown coverage values are significantly different according to our Z-score analysis (Table C.11).

When comparing the three top performing *PFASUM Search Matrices* to all *Standard Search Matrices*, we find that *PFASUM Search Matrices* deliver superior homology search performance with significantly greater coverage values (Figure 4.12) as indicated by the corresponding Z-scores (Table C.11). The highest improvements in coverage over *Standard Search Matrices* were achieved on the ASTRAL20 dataset. Similar to our findings in the evaluation based on similar entropy, this indicates that *PFASUM Search Matrices* are especially useful when searching for remote homologs.

Database	Matrix	Gap parameters	Coverage
ASTRAL20	VTML200	-14/-1	0.1598
	PFASUM60	-16/-1	0.1706
ASTRAL40	VTML200	-14/-1	0.4392
	PFASUM43	-13/-1	0.4448
ASTRAL70	VTML200	-9/-2	0.5459
	PFASUM31	-13/-2	0.5508

Table 4.7: Best performing substitution matrices of the *PFASUM Search Matrices* and *Standard Search Matrices* sets for the three test scenarios.

Surprisingly, the often used BLOSUM matrices are outperformed by VTML200 and the OPTIMA5 matrix. Additionally, BLOSUM80 – often suggested as the matrix of choice for sequence datasets with high similarity – is outperformed by BLOSUM50 and BLOSUM62 on all three test datasets. Both PAM matrices exhibit relative good performance on the high similarity dataset (ASTRAL70) but either is under-performing for sequences with more remote evolutionary relation (ASTRAL20). The MD matrices [Jones et al. 1992] deliver similar results to the lower numbered VTML matrices.

While VTML200 tends to be an universally good choice for homology search as the best performing matrix out of the *Standard Search Matrices* set on all three datasets, *PFASUM Search Matrices* can still achieve higher coverage values (Table 4.7). In general, the best performing *PFASUM Search Matrices* for the ASTRAL20, ASTRAL40, and ASTRAL70 datasets outperform all *Standard Search Matrices* on a statistical significant level (Table C.11).

Discussion

PFASUM Search Matrices perform significantly better than *Standard Search Matrices* in homologous sequence search, especially on datasets with small or limited sequence similarity such as ASTRAL20. The best performing matrix on this dataset is the PFASUM60 matrix with a relative entropy of $H = 0.4941$ bit. Interestingly, this matrix performs slightly worse on more similar datasets such as ASTRAL40 and ASTRAL70 compared to PFASUM43 and PFASUM31 which have much lower relative entropies of only $H = 0.3354$ bit and $H = 0.2297$ bit, respectively. For ungapped alignments, matrices with higher relative entropy are usually more suitable for detecting homologs within similar sequences than matrices with lower relative entropies [Altschul 1991]. This is apparently not the case when using PFASUM matrices on the tested ASTRAL datasets.

A possible explanation for these findings can be drawn from the composition of the Pfam seed alignments, the basis for *PFASUM Search Matrices*. Pfam seed alignments consist of representative sequences for each family that are aligned based on their structural properties. The sequences within a family are thus structurally similar but do not necessarily possess a similar amino acid composition. When clustering these potentially dissimilar sequences using low clustering thresholds, substitution events between them are thus attenuated. *PFASUM Search Matrices* with lower matrix number, i.e., lower clustering thresholds, apparently favor pairs of identical amino acids over substitution events and are more suited for similar sequence datasets despite their relative entropy is being small. A full assessment of this

Test set	Algorithm	Matrix numbers
PFASUM MSA Matrices	PFASUM	31, 43, 60
Standard MSA Matrices	BLOSUM	50, 62
	PAM	250
	VTML	160, 200

Table 4.8: The matrix test sets assessed in the MSA construction evaluation.

effect requires a deep and thorough analysis of the amino acid compositions in the ASTRAL dataset and Pfam seed sequences. This is, however, beyond the scope of this thesis and recommended for future research.

4.5.4 Evaluation - MSA construction

Another popular task for employing substitution matrices is the construction of Multiple Sequence Alignments (MSA). Hence, we also evaluated the impact of PFASUM matrices on the quality of MSAs using state-of-the-art MSA benchmarks. The following sections describe this evaluation in detail.

Tested substitution matrices

The calculation of pairwise sequence alignments forms the basis of many homology search tools and MSA programs. For example, the search tools SSEARCH [Pearson 1991] and BLAST [Altschul et al. 1990] employ pairwise alignments for calculating the similarity between sequences. MSA programs such as MUSCLE [Edgar 2004b] and MAFFT [Katoh et al. 2002] use pairwise alignments, e.g., during guide tree construction or when generating profile-profile alignments. This suggests that matrices which are suitable for either task may also be useful for the other task. Hence, we assess PFASUM's MSA construction capabilities by focusing on the three best performing *PFASUM Search Matrices* in our homology search performance evaluation, namely PFASUM31, PFASUM43 and PFASUM60 (Table C.1, Table C.2, and Table C.3). We refer to this matrix subset in the following as *PFASUM MSA Matrices* (Table 4.8).

Out of the set of *Standard Search Matrices*, we chose the PAM250, BLOSUM50, BLOSUM62, VMTL160, and VTML200 matrices for this evaluation. These matrices are used as default matrix by several MSA algorithm such as MUSCLE and MAFFT. We denote this matrix subset in the following as *Standard MSA Matrices* (Table 4.8).

Benchmark datasets

To compare the quality of MSAs generated using our novel PFASUM matrices with those created with conventional matrices, we used the MSA benchmark collection bench provided by Edgar [2009]. This collection of benchmark datasets consists of commonly used MSA benchmarks stored in the FASTA [Pearson and Lipman 1988] format. From these, we selected the unmodified BAliBASE 3.0 [Thompson et al. 2005], SABmark 1.65 [Van Walle et al. 2005] and OXBench [Raghava et al. 2003] benchmarks for our evaluation. Each benchmark consists of reference MSAs and corresponding unaligned sequence sets.

BAliBASE 3.0 is one of the most widely used MSA benchmarks and provides 386 MSAs categorized in five different sets. Each set represents a specific MSA use case, e.g., a set of very divergent sequences (Reference 1) or sequence families that are aligned to a distantly related sequence (Reference 2). The MSAs in each set were generated using a combination of sequence- and structure-based methods with manual refinement [Edgar 2009c]. SABmark 1.65 provides two sets of MSAs, a "Twilight Zone" set (209 MSAs) and a "Superfamilies" set (425 MSAs) which are derived from a consensus of SOFI and CE [Boutonnet et al. 1995]. While the sequences in the first set share a maximum similarity of 25%, the second set contains sequences with a maximum similarity of 50%. The underlying sequences were selected using fold information from the SCOP database and thus possess known structure. The last benchmark used in our evaluation is OXBench. It provides a set of 395 structural MSAs constructed using `STAMP` [Russell and Barton 1992] and 3D structural information from the 3Dee database [Siddiqui et al. 2001].

MSA methods

For our evaluation, we constructed 543,360 MSAs in total using the popular `MUSCLE` algorithm (v3.8.425) [Edgar 2004a, Edgar 2004b] in combination with the aforementioned substitution matrices and different gap penalties. Similar to our homology search performance evaluation, we varied the gap opening and gap extension penalties between −5 and −20 and −1 and −3, respectively, to prevent the bias from potentially inaccurate gap penalty settings.

`MUSCLE` constructs MSAs in three steps, i.e., a draft progressive, an improved progressive, and an iterative refinement step. In order to mitigate MSA quality differences solely induced by refinement steps, we set the maximum number of iterations to one. The MSAs are thus computed using a matrix independent guide tree and a single progressive alignment step only based on our chosen parameters. Hence, the quality of the generated MSAs only depends on the evaluated substitution matrix and gap penalties.

MSA quality evaluation

We measured the quality differences between our generated MSAs and the reference MSAs using the q-score measure [Edgar 2004b] implemented in the identically named tool `qscore` by Edgar [2009]. This measure describes the fraction of identically and thus correctly aligned amino acid pairs between a test and a reference MSA. In other words, the quality of a test MSA can be expressed as a number between 0 and 1.

We use the q-score in two different evaluation scenarios. First, we calculate the average q-score $\bar{q}$ over all MSAs in a benchmark dataset for each substitution matrix separately. This allows a general comparison but is obviously sensitive to strong outliers. To compensate this issue, we provide a second evaluation scenario. Here, we count the number of times that a specific PFASUM matrix in the *PFASUM MSA Matrices* set produced an MSA of at least as good quality as a specific matrix out of the *Standard MSA Matrices* set.

Results

Whereas homologous sequence search assessment aims at evaluating the performance of substitution matrices for pairwise sequence alignments, the alignment of multiple sequences in MSAs is another field of application for substitution matrices. We will first compare the average performance of matrices in the *PFASUM MSA Matrices* set to conventional

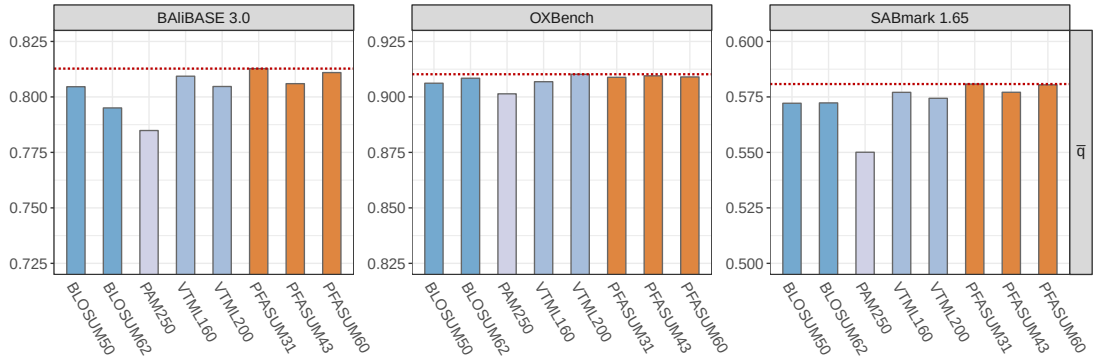


Figure 4.13: General comparison of MSA matrix performance based on the average q-score $\bar{q}$ per benchmark database. *PFASUM MSA Matrices* outperform the tested *Standard MSA Matrices* on the BAliBASE 3.0 and SABmark 1.65 benchmarks. PFASUM31 achieved the highest $\bar{q}$ for BAliBASE 3.0 and SABmark 1.65. VTML200 leads all matrices on the OXBench dataset. The red dotted line indicates the maximum $\bar{q}$ separately for each benchmark.

substitution matrices grouped in the *Standard MSA Matrices* set (Table 4.8) on three popular MSA benchmark datasets. In the second part, we will dissect these results and investigate how *PFASUM MSA Matrices* fare on single MSAs in comparison to *Standard MSA Matrices*.

a) Average matrix performance

To properly evaluate the capabilities of *PFASUM MSA Matrices* in comparison to *Standard MSA Matrices* for MSA construction, we computed MSAs based on three different MSA benchmark datasets using the MSA program `MUSCLE` in combination with the aforementioned matrices and varying gap penalties. The quality differences between these MSAs and their benchmark reference MSAs were measured afterwards using the q-score measure. Figure 4.13 shows the results for the average q-score ($\bar{q}$) for all tested matrices on the BAliBASE, OXBench and SABmark datasets.

As expected, we observed a significantly higher $\bar{q}$ for all matrices on the OXBench dataset than on BAliBASE 3.0 and SABmark since $\geq$ 73% of the MSAs in this dataset consist of sequences with at least 40% sequence identity. Contrarily, the SABmark dataset presents a decisively more difficult challenge for all matrices. Over 93% of all alignments contain sequences with less than 40% sequence similarity. The BAliBASE 3.0 dataset can be placed in between SABmark and OXBench in terms of sequence identity. At least 63% of the BAliBASE 3.0 alignments contain sequences with less then 40% similarity.

Even though we analyze the alignment quality of substantially different test datasets, we find our *PFASUM MSA Matrices* in the top three performing matrices on all datasets. On BAliBASE 3.0, PFASUM31 achieved the highest $\bar{q}$ of all matrices with $\bar{q}_{\text{PFASUM31}} = 0.8128$, closely followed by PFASUM60 with $\bar{q}_{\text{PFASUM60}} = 0.8110$ and VTML160 with $\bar{q}_{\text{VTML160}} = 0.8093$. The highest alignment quality on OXBench is reported for VTML200 (at $\bar{q}_{\text{VTML200}} = 0.9102$) only marginally besting PFASUM60 with $\bar{q}_{\text{PFASUM60}} = 0.9095$. For the SABmark dataset we find the highest two performances for PFASUM31 ($\bar{q} = 0.5808$) and PFASUM60 ($\bar{q} = 0.5804$). In this case, the next highest average alignment quality by a matrix out of the *Standard MSA Matrices* set was achieved by VTML160 ($\bar{q} = 0.5771$).

		PFASUM31	PFASUM43	PFASUM60
BAliBASE 3.0	**BLOSUM50**	67.36 (59.07)	62.69 (54.92)	62.44 (51.81)
	BLOSUM62	71.50 (65.03)	69.43 (62.44)	67.88 (58.03)
	PAM250	75.39 (70.21)	71.76 (63.73)	70.73 (66.84)
	VTML160	63.47 (54.92)	61.14 (50.52)	61.66 (48.45)
	VTML200	68.13 (57.77)	63.21 (51.81)	61.92 (50.00)
OXBench	**BLOSUM50**	81.52 (25.06)	83.29 (26.84)	84.30 (23.80)
	BLOSUM62	79.24 (23.04)	81.01 (22.03)	80.76 (21.52)
	PAM250	80.00 (32.41)	82.03 (31.65)	79.49 (33.16)
	VTML160	79.24 (24.81)	83.80 (28.61)	82.53 (25.32)
	VTML200	76.96 (20.76)	82.03 (22.53)	80.00 (20.51)
SABmark 1.65	**BLOSUM50**	67.85 (47.52)	63.36 (44.21)	65.96 (43.74)
	BLOSUM62	63.59 (48.23)	64.30 (47.28)	64.30 (45.86)
	PAM250	72.58 (59.57)	70.21 (56.97)	72.81 (60.05)
	VTML160	64.78 (45.39)	60.76 (43.50)	65.25 (42.08)
	VTML200	66.67 (47.99)	63.83 (44.21)	68.56 (44.21)

Table 4.9: Fraction of times (in percent) that a specific matrix in the *PFASUM MSA Matrices* set produced an MSA of at least as good ($\geq$) quality as a specific matrix out of the *Standard MSA Matrices* set. The comparison for better-than-relations ($>$) are shown in brackets. The values are shown for all *PFASUM MSA Matrices* vs. *Standard MSA Matrices* comparisons on all three different benchmark datasets.

The SABmark dataset allows us to delve deeper in the performance of substitution matrices on alignments with very low sequence identity ('twilight zone' dataset) and moderately difficult alignments ('superfamily' dataset). On both datasets, we observe that *PFASUM MSA Matrices* outperform the *Standard MSA Matrices* with PFASUM60 achieving the highest $\bar{q}$ for the 'twilight zone' dataset, while PFASUM31 performed the best for the 'superfamily' alignments (Table C.12).

In summary, *PFASUM MSA Matrices* outperform all analyzed *Standard MSA Matrices* on average on benchmark datasets with low sequence identity. In this case, PFASUM60 is the matrix of choice for very difficult alignments of sequences with low sequence identity. For alignments with moderate complexity and medium sequence similarity, PFASUM31 proofs to generate better alignments than any of the *Standard MSA Matrices*.

b) Quality improvements over *Standard MSA Matrices*

While the average q-score is an overall assessment of the alignment quality, directly comparing the performance between two matrices on alignments can yield insights on whether the average is dominated by strong outliers. Hence, we chose to compare *PFASUM MSA Matrices* with *Standard MSA Matrices* on a per alignment level based on their reported q-score values. For this, we count the number of times that a specific tested PFASUM matrix produced an MSA of at least as good or higher quality as a specific matrix out of the *Standard MSA Matrices* set. The results for this comparison based on BAliBASE 3.0, OXBench and SABmark are shown in percent in Table 4.9.

PFASUM MSA Matrices achieved a q-score at least as good as the *Standard MSA Matrices* in over 60% of all BAliBASE 3.0 alignments, outperforming them in at least 50% of the test cases. In comparison to PAM250, the usage of *PFASUM MSA Matrices* even resulted in higher quality in over 63% of the test cases. Similar to BAliBASE 3.0, over 60% of all SABmark alignments reconstructed using *PFASUM MSA Matrices* show a comparable quality than those generated with *Standard MSA Matrices* and at least 42% are of higher quality.

In contrast to the other two benchmarks, the performance gain of *PFASUM MSA Matrices* over *Standard MSA Matrices* on OXBench MSAs is rather small. Only 20% to 33% of the MSAs generated with *PFASUM MSA Matrices* show larger q-scores than those constructed with *Standard MSA Matrices*. However, between 76% and 84% of the PFASUM generated MSAs are at least as good as their counterparts. Interestingly, while VTML200 achieves a higher average q-score $\bar{q}$ than any of the *PFASUM MSA Matrices* on OXBench alignments, *PFASUM MSA Matrices* still produced higher or equal quality MSAs than VTML200 in over 76% of these alignments.

Discussion

Our performance evaluation shows that SSEARCH using *PFASUM Search Matrices* provides significantly better search results and as such also higher quality pairwise sequence alignments. Since these form the basis for many state-of-the-art MSA algorithms such as MUSCLE and MAFFT, we also tested the capabilities of a selection of the aforementioned matrices for MSA construction. Our results indicate that the tested *PFASUM MSA Matrices* perform exceptionally well when aligning sequences with medium to low sequence similarity such as in the BAliBASE 3.0 and SABmark 1.65 benchmarks. However, the performance differences between *PFASUM MSA Matrices* and the best performing *Standard MSA Matrices* on datasets containing similar sequences such as OXBench is rather small.

This effect can be related to compositional similarities between the matrices which in particular affects the alignment of similar sequences. All tested matrices in our MSA evaluation, with the exception of PAM250, share comparable scoring ratios between diagonal and off-diagonal entries per amino acid, favoring amino acid conservation over substitutions. Since similar sequences are usually more conserved and the majority of the matrix differences can be observed on the off-diagonal, the alignments generated with *PFASUM MSA Matrices* and *Standard MSA Matrices* only show minor differences.

4.5.5 Conclusion

We presented the novel PFASUM substitution matrices for the accurate detection of homologous protein sequences and for scoring and constructing high quality protein MSAs. Our PFASUM matrices are based on the Pfam seed dataset [Finn et al. 2016] (version 29.0) which represents the currently known sequence space covering a large variety of related and divergent sequences. The MSAs in this dataset are also manually curated by experts. Hence, the data basis for PFASUM substitution matrices is not only much larger and diverse than those of conventional substitution matrices but also represents ground truth data instead of automatically generated and thus potentially biased data. In contrast to conventional construction methods, our algorithm can also effectively handle unfiltered MSAs and ambiguous amino acid symbols and thus prevents the loss of potentially important information. An in-depth evaluation showed that these features enable PFASUM substitution matrices

to deliver significantly better homology search results and produce more accurate MSAs than conventional matrices. One of the best performing PFASUM matrix for homologous sequence search is PFASUM60, especially when searching for distantly related homologs. PFASUM60 also showed reasonable quality improvements for MSA construction. We thus recommend PFASUM60 as a general choice for these particular tasks.

Chapter 5
Visual analysis and comparison of multiple sequence alignments

5.1 Introduction

Sequence alignments, and especially multiple sequence alignments (MSA), are the basis for several important applications in modern biology such as evolutionary heritage, domain analysis, or protein structure prediction. While optimal pairwise sequence alignments (PSA) can be computed with dynamic programming in less than one minute using a standard PC (Section 2.2), the construction of an optimal multiple sequence alignment under the widely used sum-of-pairs scoring model (Section 2.4.1) corresponds to an NP-complete optimization problem [Wang and Jiang 1994, Just 2001, Elias 2006]. Thus, optimal MSAs with more than 10 sequences cannot be computed in reasonable time even on modern computers. To cope with the NP-complete nature of the MSA problem, state-of-the-art MSA algorithms typically use heuristics to speed up the computation process (Section 2.4). The usage of heuristics such as the progressive alignment method (Section 2.4.1) comes, however, at the cost of reduced or at least uncertain alignment quality. This may have a strong negative impact on all subsequent analysis and applications. For this reason, MSA quality assessment and manual MSA refinement are crucial steps to produce reliable MSAs for further analysis and research tasks (Section 1.3).

The vast number of MSA algorithms and programs that have been proposed over the last years (Section 2.4.2) and their huge parameter space is another problem that demonstrates the importance of MSA quality assessment. Choosing a suitable MSA program and especially an optimal scoring model (i.e., a substitution matrix and gap penalties) for a given alignment problem is still an unsolved problem and subject of current research [Giribet and Wheeler 1999, Reese and Pearson 2002, Price et al. 2005, Agrawal and Huang 2009, Edgar 2009c, Kececioglu and DeBlasio 2013, Hess et al. 2014a]. Notably, there is no guarantee that a particular scoring model produces evolutionary correct alignments.

As a result of these uncertainties, most users – in particular those without specific knowledge in the field of MSA – often generate MSAs by simply using well established algorithms in combination with default parameters, even though these programs and settings may be suboptimal for the particular alignment task. For instance, the enormous number of over 42,000 citations of `ClustalW` (Section 2.4.2) implies that `ClustalW` is still the most used MSA program even though it has not been consistently reported to compute the most accurate alignments [Chatzou et al. 2016]. Hence, analyzing the quality of MSAs or comparing MSAs computed with different algorithms and parameter sets may substantially help in choosing more suitable programs and parameters and thus result in better alignments.

Despite the obvious usefulness of MSA quality assessment, this aspect is generally unnoticed by the biological community [Anderson et al. 2011, Chatzou et al. 2016]. There are several reasons for this. For instance, most users lack more profound knowledge about the MSA algorithms and the impact of their parameters or simply blindly trust the chosen programs. Chatzou et al. [2016] even speculates "on the existence of a strong methodological inertia within the biological community, where tool usage tends to snowball through protocol recycling". Another problem is the lack of visual analysis and comparison tools that effectively support the intricate task of MSA quality analysis.

In this thesis part, we address the latter issue by presenting a novel interactive visual comparison and analysis approach for protein MSAs. It enables even non-expert users to visually explore, compare and analyze multiple MSAs in order to assess their quality and the impact of different MSA algorithms and scoring models. The comparison and analysis of (alternative) MSAs can be performed on global as well as on local levels supported by different highlighting techniques based on automatic quality assessment.

5.2 Related work

Like other visual analytics techniques, our interactive visual comparison and analysis approach for MSAs combines automatic data mining methods with different visualizations and interactions. For this reason, we first describe and discuss related work about methods for the automatic comparison and quality analysis of MSAs. The second part of this section, describes state-of-the-art visualization and visual comparison techniques for MSAs in detail.

5.2.1 MSA comparison and quality analysis

Due to the NP-complete nature of the MSA problem, the optimal MSA under a given scoring model is generally unknown. According to Kececioglu and DeBlasio [2013], MSA quality is thus usually assessed by two different principles. The first one is to compute a number of alternative alignments for the same set of sequences using different alignment algorithms, parameter settings, and scoring models. Afterwards, the generated alignments are analyzed by identifying consistently aligned regions over all alternative MSAs which are considered to be valid [Vingron and Argos 1990]. The second method is to infer alignment quality directly from an MSA's inherent structure by analyzing different criteria such as the number of gaps or the average symbol diversity in the MSA columns [Kececioglu and DeBlasio 2013].

MSA quality assessment based on comparison of alternative alignments

Over the last years, several methods have been proposed that apply the first principle of comparing alternative alignments to identify reliable regions [Vingron and Argos 1990, Mevissen and Vingron 1996, Vingron 1996, Cline et al. 2002, Lassmann and Sonnhammer 2005, Sela et al. 2015]. For instance, Vingron and Argos [1990] presented a scoring based approach to estimate the reliability of a specific position within a pairwise alignment. The authors compared the score of a particular PSA with the score of the optimal PSA where this specific position had been removed.

Another approach by Vingron [1996] uses *near-optimal alignment analysis* to predict reliable regions in a set of alternative PSAs. Consistently aligned regions across a set of alternative alignments are then considered to be reliable and thus correctly aligned. In contrast, those with large differences are assumed to be unreliable and potentially need further refinement.

Consistently aligned regions between two alignments can be detected, e.g., using local MSA comparison measures (Section 2.5.2). This includes measures that quantify the alignment differences on a per-column (e.g., *BAliBASE total column score* [Thompson et al. 1999a]) or on a per-residue level (e.g., q-score [Edgar 2004b]).

Unlike other local measures, the *shift score* proposed by Cline et al. [2002] also allows to quantify the magnitude of the per-residue alignment differences as explained in detail in Section 2.5.2. For each residue x, i.e., a single symbol x in a sequence, in two alternative MSAs A and B, a shift value is calculated on the basis of the residues aligned to x in these two MSAs. If x is aligned with the same residues in both MSAs, the shift is zero. Otherwise, the shift represents the difference between their indices. This shift value is then transformed by an affine function to compute the final shift score within a pre-defined scoring range. For these reasons, we use the *shift score* measure in our approach as the basis for the visual pairwise comparison of MSAs (Section 5.3.3).

MSA quality assessment based on alignment inner structure

As outlined above, another principle for MSA quality assessment is based on the direct analysis of an MSA's inner structure. Two of the most commonly used measures for the direct quality assessment of MSAs are the column based *sum-of-pairs* score [Thompson et al. 1999b] previously described in Section 2.4.1 and the *consensus* score. The former rates the quality of an alignment column by accumulating the substitution scores of its residue pairs using a particular substitution matrix such as BLOSUM62 [Henikoff and Henikoff 1992a]. The latter measures the overall diversity of residue types per column, often based on a simple majority rule with gaps optionally taken into account.

The disadvantage of both methods is that they do not take neighboring columns into account and that their scores can easily be misinterpreted. Columns containing several diverse amino acids are typically rated by these measures with low scores and are thus considered to be probably misaligned. When investigating these low-scoring columns in the context of their neighboring regions, they may still reflect correct alignments from an evolutionary perspective. Also, the scores reported by the column-wise sum-of-pairs measure typically cannot be compared in a fair way. Since most substitution matrices report different scores for preserved amino acid pairs (e.g., BLOSUM62: AA = 4 vs. WW = 11), the sum-of-pairs measure rates even fully conserved columns with different scores depending on their underlying amino acid type. Nevertheless, we adopted both measures in our approach in order to provide state-of-the-art quality criteria that can still be used for a rough assessment of alignment quality.

Recently, Kececioglu and DeBlasio [2013] presented further alignment-only based quality measures for protein MSAs. These measures were initially developed to train a parameter advisor for sequence alignment called FACET on state-of-the-art MSA benchmark datasets [Kececioglu and DeBlasio 2013]. In their report, the so-called *Average Substitution Score*, the *Gap Open Density*, and the secondary structure-based measures *Secondary Structure Blockiness*, *Secondary Structure Identity*, and *Secondary Structure Agreement* were reported to perform best. However, the usage of these measures for the quality assessment of arbitrary MSA is limited as outlined below.

The authors describe the *Average Substitution Score* as the average score of all substitutions in an alignment measured with an adapted version of the BLOSUM62 matrix [Henikoff and Henikoff 1992a]. This variation of the BLOSUM62 matrix has been shifted and scaled to a scoring range of $[0, 1]$. Hence, this measure is identical to the standard sum-of-pairs score including all aforementioned drawbacks even though its total score lies within the range of 0 to 1.

The *Gap Open Density* represents the fraction between the number of gaps and the overall gap length. In general, longer gaps are considered to be more likely than the same number of gap symbols at noncontiguous sites [Lesk 2013, p. 184]. For this reason, this measure could be used as a very rough indicator for alignment quality but severely lacks the possibility of providing normalized scoring values that can be meaningfully compared. Additionally, it only allows quality assessment on a global level.

The three secondary structure-based measures use secondary structure annotations predicted with `PSIPRED` [Jones 1999] to measure the alignment accuracy. The *Secondary Structure Blockiness* is the number of non overlapping blocks of residues with identical secondary structure annotation, while the *Secondary Structure Identity* corresponds to the fraction of identical secondary structure annotations per column. The last secondary structure-based measure called *Secondary Structure Agreement* represents the probability that two residues belong to the same secondary structure type by averaging the normalized confidence values reported by `PSIPRED` for neighboring residues. While using secondary structure information for the detection of local misalignments is generally useful, many sequence datasets do not report the required secondary structure annotations. Hence, an analysis based on predicted secondary structure always introduces an additional bias and also comes at the cost of increased computation time.

In summary, even though some of the aforementioned direct MSA quality measures may be useful to identify probably correctly aligned regions, most of them are still not able to reliably identify misaligned regions. This further highlights the importance of visual analysis tools for sequence alignments since humans can effectively detect misalignments through their visual system and natural pattern recognition (Section 3.5).

5.2.2 Visual analysis and comparison of MSAs

Many different MSA visualization tools have been published to this day (e.g., `ClustalX2` [Larkin et al. 2007], `Jalview` [Waterhouse et al. 2009], `SeaView` [Gouy et al. 2010], `Webprank` [Löytynoja and Goldman 2010], `SuiteMSA` [Anderson et al. 2011], `SBAL` [Wang et al. 2012], `AliView` [Larsson 2014]). Some of these tools also provide additional functionality such as manual MSA editing and interfaces to prominent alignment algorithms and web services for sequence retrieval or secondary structure prediction. In the following, we describe a representative selection of these programs and approaches in more detail.

Jalview

One of the most prominent visual editors for multiple sequence alignments is the Java-based program `Jalview` [Waterhouse et al. 2009]. Like other MSA visualization programs, `Jalview` depicts an MSA in the form of a grid filled with the one letter codes of the residues or gap symbols. The background of each grid cell can be colored according to specific criteria such as the physico-chemical attributes of the corresponding residues. This allows the visual

detection of potential misalignments or the identification of interesting alignment regions. However, the shape of the residues cannot be adapted in order to transport additional information on a second visual dimension like the representation used in our citizen science game *Bionigma* (Section 3.5).

Below the alignment visualization, further information about the different alignment columns is shown. For example, this includes the consensus amino acid type, the degree of amino acid conservation, the fraction of non-gap symbols, and a column quality indicator based on BLOSUM62 scores. As mentioned above, even though these measures may be useful for the detection of potentially correctly aligned regions, they still do not take neighboring acids or columns into account and cannot reliably detect misaligned regions. Hence, quality assessment in `Jalview` primarily relies on the visual detection of misalignments through color encoded residues and pattern recognition.

In addition, `Jalview` does not have a zooming functionality and instead only relies on a one-pixel-per-residue overview visualization. This severely limits its usefulness for the analysis of larger sequence alignments. However, `Jalview` enables the user to hide specific regions of the MSA. The shown MSAs can be manually refined and annotated through several editing functions including copy-&-paste of sequences or sequence segments and shifting of residues.

Other features of `Jalview` include access to web services for sequence alignment and secondary structure prediction as well as functions for deriving and inspecting phylogenetic trees. If available, `Jalview` also supports the visual exploration of a protein's 3D structure using Jmol[1] interlinked with the corresponding 2D MSA visualization.

In summary, `Jalview` is a feature-rich MSA editor but its capabilities for the visual quality assessment of MSA are limited, especially for larger MSAs. Additionally, even though `Jalview` supports visualizing multiple MSAs in separate windows, it does not provide any functionality for comparing alternative alignments.

AliView

`AliView` [Larsson 2014] is a light-weight visual editor for multiple sequence alignments which focuses on fast data processing and fluid visualization of large MSAs. To achieve these goals, `AliView` creates an index of the sequences in the files and only caches them in memory when they are actually viewed as well as employs a multi-threaded method for MSA rendering. Like `Jalview`, `AliView` is written in Java and visualizes MSAs in the form of a grid filled with the one letter codes of the residues. `AliView` also supports several color schemes to visually encode similarities between the residues. However, the shape and texture of the residues cannot be changed and there is also no possibility to individually customize the color schemes. This may limit the user in performing more specialized analysis tasks.

For the quality analysis of MSAs, `AliView` provides a small set of highlighting mechanisms. This includes, e.g., highlighting residues that differ from the consensus of an MSA column or that belong to it, and highlighting of residues that deviate from a particular "trace" sequence. Again, these methods may support the user in detecting potentially well-aligned regions but still restrict the analysis to single columns and do not reliably reveal misaligned regions.

[1] Jmol: an open-source Java viewer for chemical structures in 3D. `http://www.jmol.org/`, last accessed 18.10.2017.

However, AliView provides the user with an unlimited zooming function. Through this, one can easily get an overview of the entire alignment as well as inspect specific alignment regions in detail on demand. This function in combination with the fast MSA rendering method used by AliView allows the user to effectively visualize even very large MSAs.

MSAs can be edited in AliView using standard functions such as insertion or deletion of sequences or shifting sequence segments. In addition, AliView allows to merge overlapping sequences into a consensus sequence, provides an unique functionality to compute all possible primers for a user-selected alignment region, and enables automatic re-alignment of MSAs or regions through external MSA programs.

AliView's main advantages are its lightweight design and fast visualization of very large MSAs. However, its capabilities for the visual analysis of alignment quality are limited through the lack of highlighting mechanisms and the missing functionality of comparing alternative MSAs.

SuiteMSA

SuiteMSA [Anderson et al. 2011] is a collection of different visual analysis tools for multiple sequence alignments and related fields. This includes, e.g., an MSA Viewer to visualize single MSAs, a Phylogeny Viewer to view and edit phylogenetic trees, and a separate graphical interface to execute external MSA programs and to control their parameters. Additionally, SuiteMSA enables the rare option to visually compare MSAs through the programs MSA Comparator and Pixel Plot.

In the MSA Viewer, the MSA is depicted similar to the aforementioned programs using a grid-like view containing the one letter codes of the residues. Likewise, the residues can be colored according to pre-defined color schemes in order to emphasize similarities and differences between them. Besides these color schemes, the appearance of the residues cannot be further adapted, e.g., by user-customized colors or shapes which may restrict the user in the analysis. Also, MSA Viewer only provides very limited options for the quality analysis of MSAs. This includes the possibility to visualize separately obtained structure information in the form of an additional MSA shown below the original one and a bar chart indicating the level of column conservation. Since MSA Viewer does neither provide a zooming function like that in AliView nor an overview visualization as in Jalview, larger MSAs cannot be effectively analyzed, especially when showing additional structural information below the MSA. The manual editing functionality of MSA Viewer is also very limited by only allowing the deletion and insertion of gaps instead of moving individual residues or sequence segments which would be substantially more intuitive.

As mentioned above, SuiteMSA also contains programs for the visual comparison of MSAs. For instance, the MSA Comparator allows the visual comparison of a reference MSA shown at the top and an alternative MSA depicted at the bottom. Between both visualizations, the column-wise sum-of-pairs score of the corresponding alignment columns can be optionally shown in order to identify potentially similar columns. By selecting a specific column range in the reference MSA, the respective residues in the alternative alignment are highlighted. Consistently aligned residues are highlighted in blue, while differently aligned residues are marked in red. This allows the direct visual assessment of alignment differences and similarities. However, since the color property is used for encoding alignment consistency

and no further color scheme can be applied, the inner structure of the alignments can barely be assessed. Also, `MSA Comparator` neither allows MSA editing nor does it provide a zooming function. This effectively restricts its usage to the comparison of relatively small alignments.

The `Pixel Plot` enables the rudimentary comparison of more than two MSAs. The different MSAs visualizations are shown in the form of a list. Each residue is represented as a small black square on a white background. Similar to `MSA Comparator`, one can select a specific column range in the reference alignment shown at the top of the list which results in the corresponding residues being highlighted in the alternative MSAs. However, unlike the method provided by `MSA Comparator`, the `Pixel Plot` does not color-code consistently or differently aligned residues which severely limits its usefulness for the comparison of alignments. The missing editing and zooming functionality additionally amplify this limitation.

Even though `SuiteMSA` provides a number of tools for the visual analysis and comparison of MSAs, most programs have severe limitations. Larger MSAs cannot be effectively analyzed due to the lack of a zooming function and accessing all features of `SuiteMSA` requires the usage of different programs that are not connected. For instance, it is neither possible to edit an MSA in other programs than the `MSA Viewer` nor to directly visualize these changes in already running instances of `MSA Comparator` since the `SuiteMSA` programs do not synchronize their data.

Summary

Most of the aforementioned tools only provide limited capabilities to visually analyze an alignment's quality or to compare two or more MSAs. They only rely on *consensus* or *sum-of-pairs* scores which have several limitations and drawbacks. Additionally, they do not support the comparison of MSAs or only in a very limited way.

To address these issues, our approach provides a broader range of measures that are used to directly present quality information inside the MSA visualization supported by various highlighting techniques. Our method also enables quality assessment on global as well as on local levels, allows to focus on similarities or differences depending on the user's demand, and enables the analysis and comparison of multiple large MSAs. In the following section, we present our approach for the visual analysis and comparison of sequence alignments in detail.

5.3 Approach

Typical analysis tasks to assess the quality of an MSA include the identification of consistently and differently aligned regions in a set of alternative MSAs and the analysis of the inherent structure of an MSA, e.g., by identifying conserved columns and local misalignments (Section 5.2.1). Existing approaches are constrained by several limitations as discussed in Section 5.2 but also provide useful features. For this reason, we developed a novel approach for the visual analysis and comparison of sequence alignments. It combines the strengths of existing concepts with new features to overcome the limitations mentioned in Section 5.2. Our approach provides the following core features:

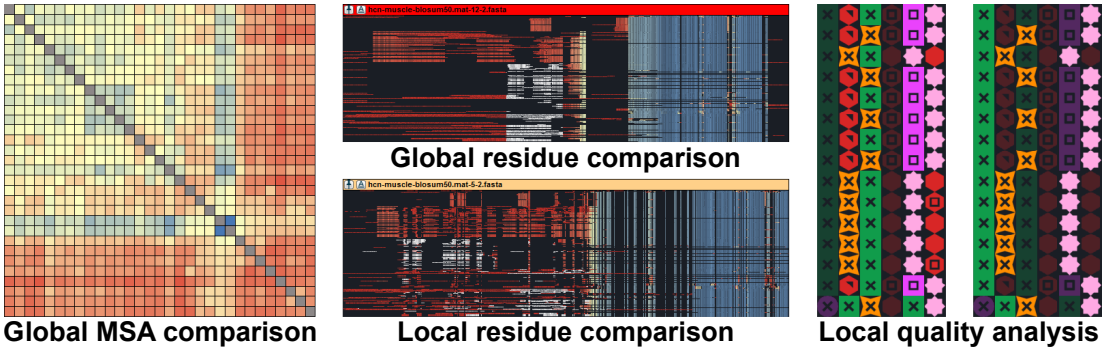


Figure 5.1: Our approach allows a novel visual comparison and analysis of multiple alternative MSAs on different levels of detail. Left: A Similarity Matrix depicting the pairwise MSA similarities of the dataset representing global comparison information. Center: Comparison of two large MSAs. For a local comparison, the MSA at the bottom highlights differently (red), slightly different (yellow) and consistently aligned residues (blue) with respect to the reference MSA shown at the top. In contrast, the reference MSA at the top shows this information globally with respect to all MSAs in the dataset. White highlights depict MSA differences for a specific selection. Right: Different highlighting modes allow the detection of potentially misaligned regions and local assessment of alignment quality.

- Our analysis system can handle multiple large MSAs simultaneously through the usage of multi-threaded rendering combined with caching and an unlimited zooming function inspired by `AliView` [Larsson 2014].

- The MSA visualizations can be fully adapted to the user's needs by either choosing a predefined visualization theme or by customizing the shape, the texture, and the color of the individual amino acid types. Thus, each visual property can be used to encode a separate per-residue information such as different physico-chemical attributes (Figure 5.1, right).

- MSAs can be easily edited by selecting and shifting specific alignment regions to the left or the right using the arrow keys on the keyboard. Two different movement modes either allow to move neighboring acids upon collision with the selected residues or to preserve their alignment.

- Multiple alternative MSAs can be simultaneously compared on a global as well as on a local level using novel comparison measures and a visual interactive N:N and 1:N comparison style. The former allows to assess the overall similarity of the MSAs in the dataset (Figure 5.1, left), while the latter compares all MSAs in the dataset to a user-specified reference alignment on a per-residue level (Figure 5.1, center).

- Several highlighting modes enable the user to focus on specific analysis aspects such as similarities and dissimilarities between alternative alignments or within single MSAs (Figure 5.1, right).

In the following subsection, we will first describe the different quality and comparison measures employed in our approach as the basis for the different visualizations. Subsequently, we present our interactive MSA visualization used for the visual analysis and manual refinement of individual MSAs. The remainder of this section presents our concept for the visual interactive N:N and 1:N comparison of sequence alignments in detail.

5.3.1 Quality and comparison measures

Our system uses several MSA quality and comparison measures in combination with different visualizations to support the user's analysis task. These measures can be either used for local analysis, i.e., on a per-column or per-residue level and/or on a global level covering an entire MSA. Notably, the underlying measure system of our approach is modularized in order to provide an easy way to extend existing measures or to integrate new ones.

Comparison measures

One focus of our approach is the visual comparison of alternative MSAs. This requires particular comparison measures that measure alignment differences on a local and global scale. As described in Section 2.5.2, a well-established scoring measure for the comparison of two sequence alignments is the *shift score* by Cline et al. [2002]. It allows an accurate assessment of local alignment differences and similarities on a per-residue level. Unlike other comparison measures such as the q-score, Modeler score, or Developer score, this measure also takes the magnitude of the per-residue alignment differences into account. For this reason, we use the shift score as the basis for our visual MSA comparison and to derive additional comparison measures.

Notably, the shift score encodes the alignment difference (shift) of a particular residue x in a sequence S_x in relation to its aligned residues y_1 and y_2 in sequence S_y in two alternative alignments (Section 2.5.2). By computing the average of all shift scores $\Delta(x)$ obtained for a single residue x in alignment A with respect to an alignment B, one can obtain an indicator for its total alignment difference between A and B. We denote this measure as *Average Residue Shift* and use it to visualize per-residue differences between two MSAs.

Likewise, the average of the *Average Residue Shift* scores obtained for all residues in the comparison of two alignments A and B represents a global similarity score for this comparison. We denote this measure as *MSA Shift* and use it for the global pairwise comparison of two alternative MSAs.

Quality measures

Like existing visual MSA editors, our goal is to support the user's analysis task by automatic MSA quality measures. For this purpose, we provide two state-of-the-art measures: the column-wise *sum-of-pairs score* (SP) [Thompson et al. 1999b] (Section 2.4.1) and a consensus-style measure denoted as *Symbol Identity*. In contrast to other visual MSA tools, we enable the user to select a custom substitution matrix for the computation of the SP score. The sum of all column SP scores is then used as global MSA measure.

The column-wise *Symbol Identity* represents the fraction of amino acids that belong to the dominating equivalence class in the respective column inspired by the *Amino Acid Identity* measure proposed by Kececioglu and DeBlasio [2013]. Each equivalence class represents

a group of amino acids, e.g., those that share specific chemical attributes. The user can either choose from predefined sets of equivalence classes representing reduced amino acid alphabets or define new ones.

These measures allow an easy but rough assessment of alignment quality as outlined in Section 5.2.1. They are also the state-of-the-art methods for assessing the quality of MSAs and can be used to identify potentially well aligned columns. In addition, we propose a new quality measure denoted as *Global Residue Shift* based on the *Average Residue Shift* measure described above. This measure represents the average score of all *Average Residue Shifts* obtained for a specific residue in the comparison of all alternative alignments in the dataset. A *Global Residue Shift* of 1 thus demonstrates that the particular residue is consistently aligned in all alternative MSAs of the dataset. In contrast, negative *Global Residue Shifts* indicate that the corresponding residues are differently aligned in all MSAs in most cases. This enables the user to effectively determine consistently aligned regions on a per-residue level in all alternative MSAs.

5.3.2 Visual analysis of multiple sequence alignments

MSA visualization

Similar to existing visual MSA editors, our approach presents an MSA in a grid-like manner where each row corresponds to a specific sequence and the cells either correspond to a specific residue in this sequence or a gap symbol. However, since similarities between amino acids such as similar physico-chemical attributes play a central role in assessing an MSA's quality, our approach uses multiple visual properties – shape, color, and texture – for the visualization of the individual amino acid types (Figure 5.2).

Depending on the user's choice, each residue is either depicted using a colored rectangular shape like in other methods or a specific colored shape, e.g., a circular disc or star. Optionally, each residue can be either lettered with its one letter code or shown with a specific texture, e.g., a cross or small triangle. This allows to simultaneously encode different amino acid properties using the three separate visual properties, i.e., shape, color, and texture.

For instance, the visualization on the left side of Figure 5.2 uses colored circular discs to visually encode whether a specific residue is either hydrophobic or hydrophilic. The gray stars represent residues where this particular chemical property is not set. In contrast, the visualization on the right side of Figure 5.2 applies the BLOSUM62 residue theme used in our citizen science game *Bionigma* (Section 3.5).

This residue theme emphasizes amino acid similarities represented by high substitution scores in the BLOSUM62 matrix [Henikoff and Henikoff 1992a]. For instance, substitutions between the aliphatic amino acid types I, L, M, and V are rated in BLOSUM62 with high log-odds scores. In contrast, substitutions between other acid types and aliphatic amino acids are penalized or receive scores of zero. To visually encode these similarities, I, L, M, and V residues are colored in red with a hexagonal shape, while the small black textures encode the respective amino acid type. Likewise, the similarity between asparagine (N) and aspartic acid (D) is encoded by the pink color. However, their shapes are different since N is also similar to serine (S) and D is also similar to glutamic acid (E), but N is not similar to E. Likewise, D is not similar to S. Hence, the more visual properties between two residues are identical, the higher is their substitution score in the BLOSUM62 matrix.

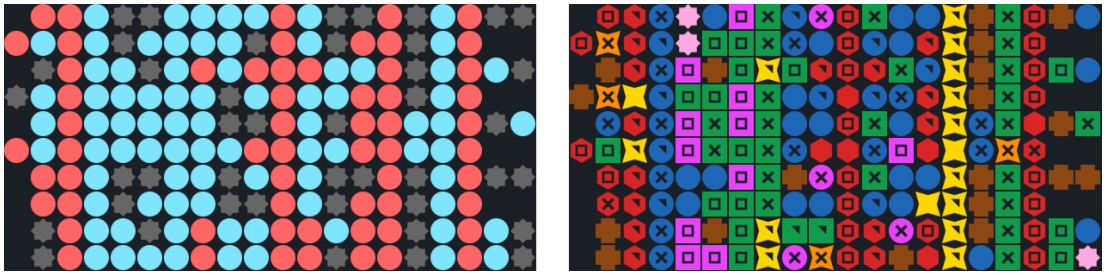


Figure 5.2: Illustration of our MSA visualization approach of a small MSA region. Left: Our residue theme emphasizing hydrophilic (cyan) and hydrophobic (red) amino acids. Other amino acids are shown as dark gray stars in order to prevent visual distraction. Right: The same MSA region using our BLOSUM62 residue theme which visually emphasizes the amino acid similarity information encoded in the BLOSUM62 substitution matrix [Henikoff and Henikoff 1992a]. The more visual properties between two residue types are identical, the higher is their substitution score in the BLOSUM62 matrix. For this reason, the aliphatic amino acids I, L, M, and V are colored in red using hexagonal shapes. Their textures indicate their amino acid types. While the color of a group is unique and thus represents the most important visual similarity property, the shape is also used to indicate weaker similarities across groups. For example, the pink and green squares represent acids that belong to different groups but still share some similarities.

In order to adjust the visualization according to the user's needs, the appearance of each individual amino acid type can be fully adapted. Additionally, we provide a set of predefined *residue themes* for common analysis tasks such as the themes shown in Figure 5.2. Besides the appearance of the residues, the user can also freely adjust the grid spacing of the alignment visualization.

To handle large MSAs effectively, our approach provides an unlimited zooming functionality inspired by `AliView` [Larsson 2014]. The user can easily zoom into the alignment at the current mouse cursor position by using the scroll wheel or zoom out to view the full alignment. In the lowest zoom level, each residue is visualized with the smallest possible size, i.e., one pixel per residue.

MSA editing

For the basic manipulation of MSAs, our approach provides several selection interactions and different movement modes (*shift* and *push*) to change the alignment inspired by our citizen science game Bionigma. The user can either select a particular residue in the alignment, specific residues in the same column, or a whole alignment region. Individual residues in the same column can be selected by left clicks, while a double-click adds all residues of the same amino acid type in this column to the selection. Regions can be selected by drawing a selection rectangle with the mouse through a drag gesture.

The selected amino acids (and gaps in case of a region selection) can then be moved to the right or left side by using the arrow keys on the keyboard. By holding a modifier key such as the shift key, the user can switch between the two provided movement modes. In the *shift mode*, the selected residues are moved in the desired direction until one of the residues "collides" with another residue that is not part of the current selection. This mode allows to

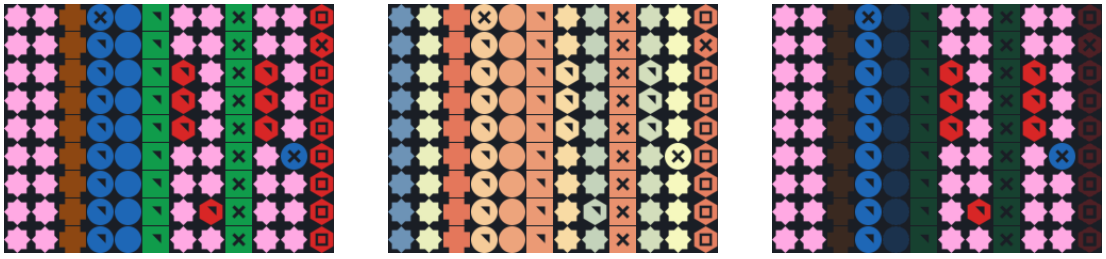


Figure 5.3: Illustration of the Symbol identity overlay and the two different overlay modes used in our approach (color and transparency). The Symbol identity measure was parameterized to rate the degree of conserved amino acids in the columns. Shown is a small alignment region of an MSA of 211 sequences in total. Left: The example alignment region visualized with the BLOSUM62 residue scheme with no overlay activated. Middle: Quality visualization of the same region using the *color mode* and a red-yellow-blue colormap. Blue colors indicate highly conserved columns while those highlighted in red depict columns of diverse amino acid types. Right: The quality visualization using the *transparency mode*. Columns with scores < 0.35 are filtered to focus on relatively conserved columns only.

manipulate the alignment locally without affecting the alignment of neighboring regions. When using the *push mode*, colliding residues are also moved in the movement direction. This allows to manipulate the alignment without restrictions.

Visual quality analysis

For the visual quality assessment of single MSAs based on their inherent inner structure, we provide the user with different *overlays*. These overlays visualize the quality of the aligned residues or alignment columns directly in the MSA visualization either using color or transparency (Figure 5.3). In the *color mode*, the alignment quality of the residues or columns are represented by colors obtained from a user-specified colormap. The alignment region shown on the left side of Figure 5.3 depicts the quality rating of the *Symbol Identity* measure using the red-yellow-blue colormap. In this case, each individual amino acid type was assigned to its own equivalence class. The Symbol Identity measure thus simply represents the fraction of conserved amino acids in each columns. Highly conserved columns are highlighted in blue, while columns with diverse amino acid compositions are marked in red.

The *transparency mode* shown on the right hand side of Figure 5.3 allows an alternative visualization of the chosen quality criteria. By adjusting a scoring threshold (≥ 0.35 in this example), the user can interactively filter the shown residues or columns to focus on those objects that represent a certain quality level either greater equal or less equal the chosen threshold. Notably, filtered objects are not fully hidden but are rendered with transparent colors. Through this, the user can maintain the context of the alignment regions while still focusing on high or low quality alignment regions.

5.3.3 Visual comparison of multiple sequence alignments

Our approach aims to enable the visual comparison of multiple alternative MSAs on global as well as on local levels. We realize this through several comparison measures presented in Section 5.3.1 in combination with different visualizations. On one hand, this includes a similarity matrix visualization for the N:N comparison of all alternative alignments. On

the other hand, this includes interlinked MSA visualizations for the 1:N comparison of an user-specified reference alignment with all other alignments in the datasets. Figure 5.4 shows our visual interactive comparison and analysis system in detail.

Similarity matrix

The N:N comparison of the alignments in the dataset in the form of a sortable *similarity matrix* visualization (1) (Figure 5.4) enables an initial visual assessment of the relative MSA similarity in the dataset. For instance, this enables a global scale analysis of the impact of different MSA algorithms or scoring parameters (e.g., different substitution matrices and gap penalties) on the resulting MSAs. Each cell in this matrix represents the similarity of two specific MSAs according to a user-specified global MSA comparison measure, e.g., the MSA Shift score measure presented in Section 5.3.1. The score obtained for each comparison is mapped to a particular color retrieved from a user-specified colormap. Optionally, the score can be normalized in order to emphasize small differences between the values.

In Figure 5.4, the red-yellow-blue colormap is selected with the MSA Shift score chosen as the comparison measure. Matrix cells representing the pairwise comparison of similar MSAs – i.e., those with MSA Shift scores near 1.0 – are thus highlighted in blue tones. In contrast, yellow-colored cells indicate MSAs with slight differences, while those highlighted in red represent MSAs with largely different structure.

Further details about individual pairwise MSA comparisons such as the corresponding MSAs and comparison scores are shown in a tooltip. It is activated by moving the mouse cursor over the cells. Additionally, the user can sort the matrix entries either alphabetically or by the comparison score. He or she can also opens a particular pairwise MSA comparison by double-clicking on the corresponding matrix cell.

MSA list

The *MSA list* (2) (Figure 5.4) shown below the similarity matrix lists all alternative MSAs in the dataset. It allows to clone MSAs to add additional alternative MSAs to the dataset. Likewise, MSAs can be removed from the dataset, selected as reference for the 1:N comparison, or shown in an MSA visualization. This list can also be sorted according to a user-specified MSA quality measure (Section 5.3.1). This allows the user to get a quick overview of the overall quality in the dataset and to select potentially interesting MSAs.

MSA comparison panel

The MSA comparison panel (3) (Figure 5.4) depicted in the center of our visual comparison system realizes the 1:N comparison feature. The MSA visualization shown at the top always refers to the chosen reference alignment to which all other alternative alignments in the dataset are compared. Residues in the reference alignment are colored according to their *Global Residue Shift* score and the chosen colormap. This enables the user to assess local residue differences on a global scale. As described in Section 5.3.1, this measure represents a residue's global alignment consistency across all MSAs in the dataset. According to the red-yellow-blue colormap used in Figure 5.4, the blue-highlighted residues in the center of the reference MSA indicate regions that are consistently aligned in the entire dataset and thus can be considered to probably be correctly aligned. In contrast, regions highlighted in red represent strongly deviating alignment regions that need further inspection.

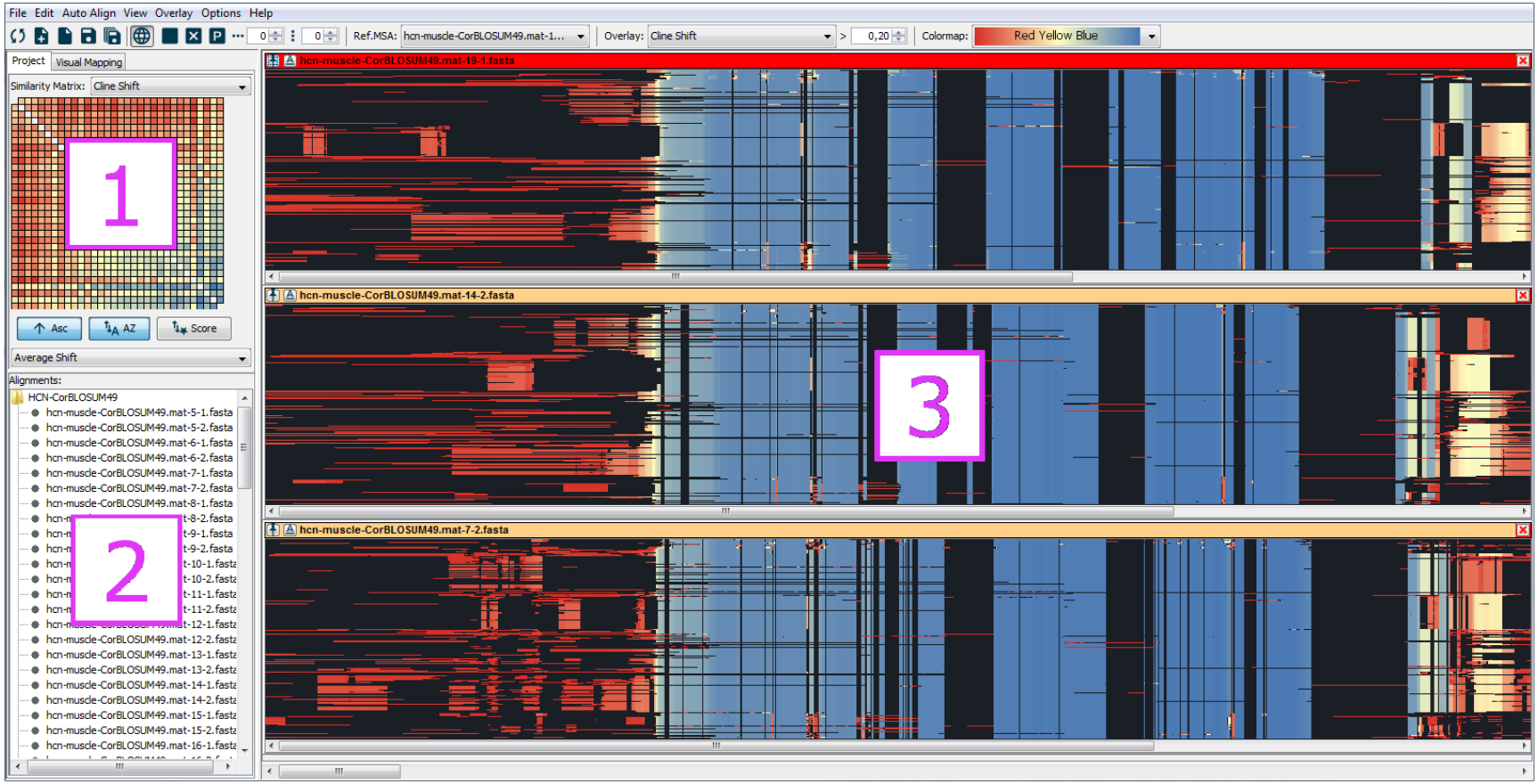


Figure 5.4: Overview of our visual comparison approach: (1) The sortable *similarity matrix* giving an overview of all pairwise MSA comparisons based on a user-selectable MSA similarity measure and colormap. (2) The *MSA list* listing all MSAs in the dataset either in alphabetical order or ordered by the score of user-specified MSA quality measure.(3) The *MSA comparison panel* showing three different interlinked *MSA visualizations*. The chosen reference MSA for the 1:N comparison is shown at the top. It highlights consistently aligned regions (blue) on a global level, i.e., over the entire dataset. The other views highlight consistently aligned regions with respect to the chosen reference MSA.

The visualizations of the alternative MSAs shown below the reference MSA represent a different comparison measure using the same colormap. Here, the highlighted per-residue alignment differences refer to the *Average Residue Shift* scores obtained for the pairwise comparison of the alternative MSA and the currently chosen reference MSA. This allows to visually assess local alignment differences between a specific alternative MSA and the reference alignment.

Notably, the user can decide to either show MSAs in the comparison panel or in separate windows to visually compare even more MSAs. Additionally, all MSA visualizations are connected to each other. For instance, this allows the user to either zoom and scroll all MSAs simultaneously or to control them separately. The main advantage of the interlinked MSA visualizations is the possibility to select and highlight specific residues or regions across all MSAs. Through this, the different alignment constellations of user-specified residues can be exactly determined.

Another interesting aspect in the analysis of the per-residue alignment differences is the magnitude of the individual shifts. To support this analysis on a more detailed level, the user can set a threshold to filter residues with shift values less equal or greater equal than a specific threshold, similar to the method used for the quality assessment of individual MSAs (Section 5.3.2).

In summary, our visual MSA comparison system allows a fast detection of consistently aligned regions on a global level – i.e., over the entire dataset – and locally with respect to the selected reference MSA. Since consistently aligned regions over a large set of alignments are usually assumed to be correctly aligned [Vingron and Argos 1990], this allows to effectively judge alignment quality on a global and local level.

5.4 Evaluation

A common problem when constructing multiple sequence alignments is the selection of a suitable alignment algorithm and corresponding parameters for the given set of sequences. One option to address this problem is to compute several alternative alignments of the same sequence set using different algorithms and parameter settings and to select the most reliable alignment for further processing.

For the evaluation of our visual analysis and comparison approach, we simulated this scenario by analyzing and comparing 8 alternative MSAs of a set of 211 HCN (Hyperpolarization-activated cyclic nucleotide-gated) ion channel proteins generated with different algorithms and corresponding default parameters. HCN ion channels are important for the functionality of cells in the heart. Hence, we address current research interests in biomedicine. To obtain our 211 sequences, we run a `BLAST` search (Section 2.3.2) using as a query the known sequence `gi355749904`.The *E*-value threshold was set to 0.00001. In addition, we kept only those sequences annotated as "hyperpolarization" and "cyclic". In a third step, a molecular biologist and bioinformatician deleted thirteen additional sequences as non-hits. In total, we obtained a set of 211 valid HCN sequences [Hess et al. 2014a].

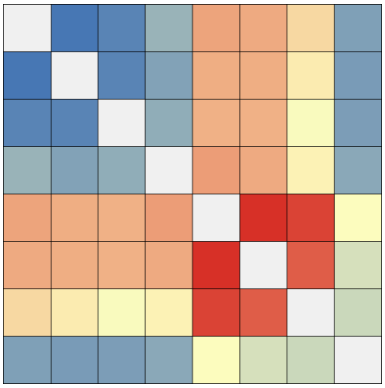


Figure 5.5: The similarity matrix view depicting the pairwise comparison of the eight MSAs computed with the different algorithms and respective default parameters. The algorithms represented by the columns from the left to the right are the four MAFFT variants G-INS-i, L-INS-i, E-INS-i, and Default, followed by MUSCLE, ClustalW2, Kalign, and ClustalOmega. Low normalized MSA Shift scores are colored in red, while those near 1.0 are highlighted in blue. Yellow tones indicate moderate MSA Shifts of ~ 0.5.

The 8 different alternative alignments of this sequence set were generated with the state-of-the-art MSA programs ClustalW2 [Larkin et al. 2007], ClustalOmega [Sievers et al. 2011], MUSCLE [Edgar 2004b], Kalign [Lassmann et al. 2009], and MAFFT [Katoh et al. 2002] using their respective default parameter settings. For MAFFT, we additionally employed its three "high-accuracy" variants L-INS-i, E-INS-i, and G-INS-i.

5.4.1 Visual comparison and analysis of HCN MSAs

In order to detect differences between the generated MSAs on a global level, we first compared them using our similarity matrix view. The overall reported similarity between the MSAs was relatively high, with MSA Shift scores of ≥ 0.9. This indicates that all tested algorithms generated relatively similar alignments. In order to visually emphasize the differences between these scores, we switched the similarity matrix view to show the min/max normalized scores (Figure 5.5).

Not surprisingly, the four generated MAFFT alignments are quite similar (columns 1 to 4), with the Default MAFFT MSA showing the most differences in this set. While these four MSAs are also structurally similar to the ClustalOmega alignment (column 8), they differ to a greater extend from the MSAs computed with MUSCLE, ClustalW2, and Kalign (columns 5 to 7). For these pairwise comparisons, our system reported normalized MSA Shift scores of up to ~ 0.32 for MUSCLE and ClustalW2, and ~ 0.52 for Kalign, respectively. The pairwise comparison of the MUSCLE, ClustalW2, and Kalign alignments among themselves (red cells in columns 5 to 7) reveal the largest differences in the dataset with normalized MSA Shift scores of < 0.12. Interestingly, all three alignments share the most similarities with the ClustalOmega MSA (column 8) which thus represents some sort of consensus alignment of all alternative alignments in the dataset. This indicates that the usage of ClustalOmega with default parameters seems to be a generally good choice for the alignment of the 211 HCN sequences used in this test scenario.

Nevertheless, the existing differences between the generated alignments should be analyzed in more detail to retrieve more profound information about the impact of the different algorithms on the resulting MSA structure. For this reason, we compared the different alternative MSAs with the ClustalOmega MSA selected as reference. The result of this comparison is shown for the N-terminal regions of the MSAs in Figure 5.6. From the top to the bottom, the shown visualizations refer to the alternative alignments generated with ClustalOmega, MAFFT-Default, G-INS-i, L-INS-i, E-INS-i, MUSCLE, ClustalW2, and Kalign.

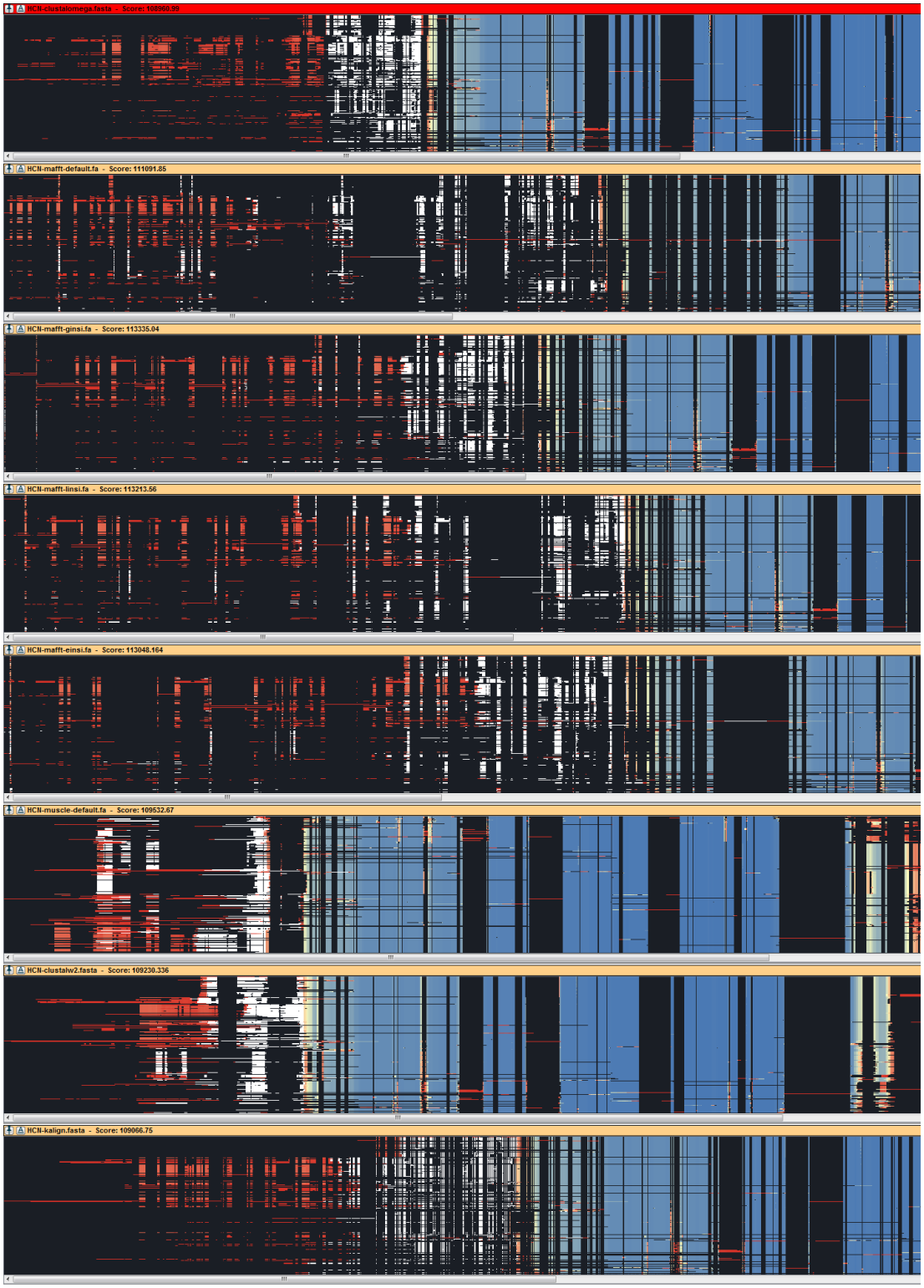


Figure 5.6: Comparison of the N-terminal region of the `ClustalOmega` MSA chosen as reference (top) and all other alternative MSAs (from top to bottom: `MAFFT-Default`, `G-INS-i`, `L-INS-i`, `E-INS-i`, `MUSCLE`, `ClustalW2`, and `Kalign`). White highlights show the different alignment constellations of selected residues.

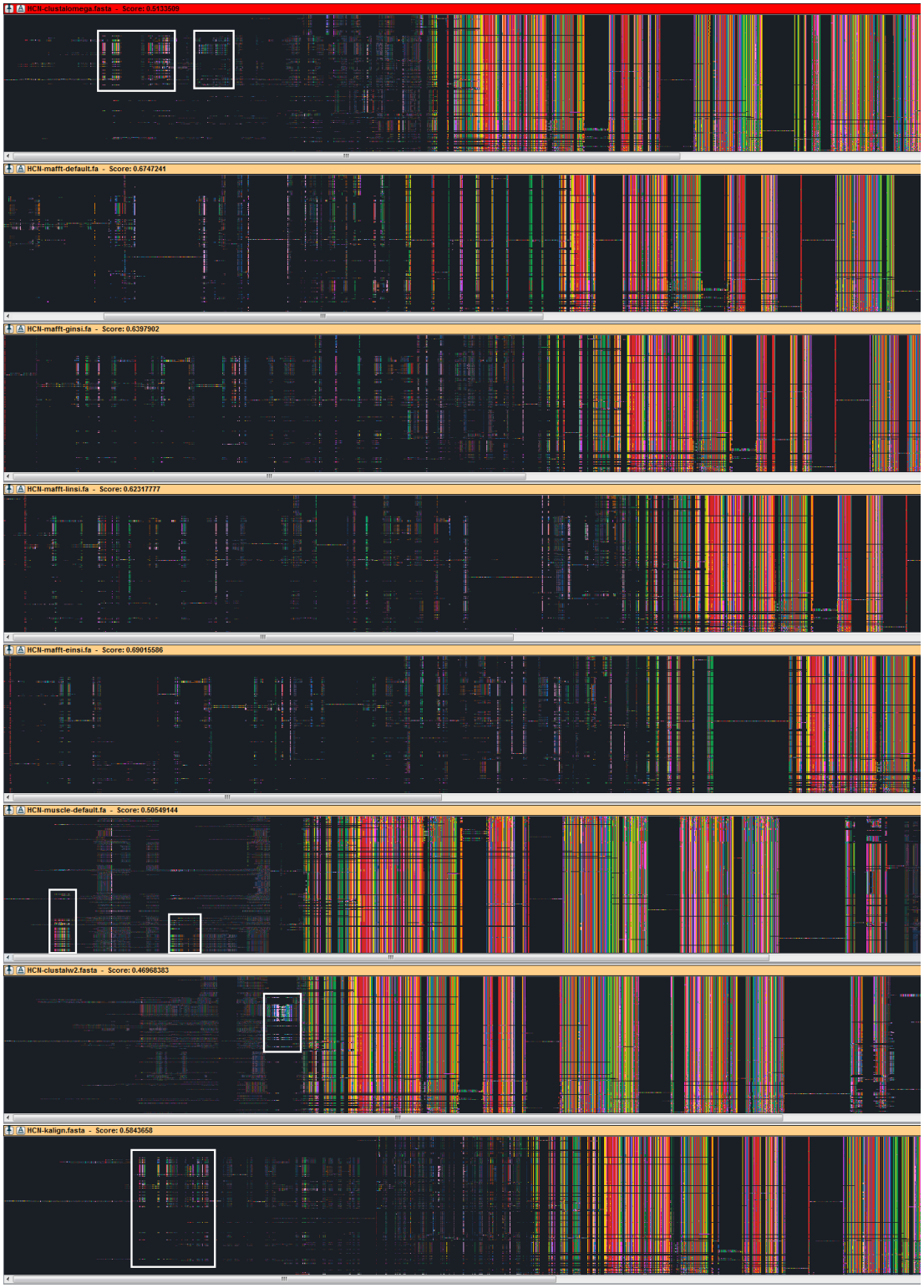


Figure 5.7: Conserved columns with Symbol Identity scores ≥ 0.40 in the N-terminal region of the alternative alignments. The reference MSA generated with `ClustalOmega` is shown at the top followed by the alternative MSAs generated with `MAFFT-Default`, `G-INS-i`, `L-INS-i`, `E-INS-i`, `MUSCLE`, `ClustalW2`, and `Kalign`.

The visualization of the reference MSA (ClustalOmega, top) using our Global Residue Shift measure reveals large regions in the center of the alignment that are consistently aligned in all alternative MSAs (blue highlights). These regions mostly refer to conserved alignment areas that were almost identically aligned by all tested algorithm (Figure 5.7). In the N-terminal area, however, the alternative alignments substantially differ as depicted by red-colored regions in the reference MSA visualization. In order to investigate these differences in more detail, we additionally selected some residues in the reference alignment (white) which highlighted their corresponding counterparts in the alternative MSAs.

As shown in Figure 5.6 (rows 2 to 5), the four different variations of MAFFT introduced more and much longer gaps in the N-Terminal regions than the other algorithms, resulting in significantly fewer mismatches but many very small "domains". When analyzing the same regions with our Symbol Identity measure using a scoring threshold of ≥ 0.40 (Figure 5.7), one can see that these small "domains" are also not well conserved. Hence, we do not see a clear motivation for the introduction of so many gaps. This is why we rate the MAFFT alignments worse than the other alignments.

Kalign (bottom) introduced many shorter gaps also resulting in several smaller domains. In contrast, the MSAs generated with ClustalOmega (top) and especially MUSCLE and ClustalW (rows 6 and 7) are more compact. As shown by the white boxes in Figure 5.7, these MSAs further generated different conserved regions which cannot be directly observed in the MAFFT MSAs. After investigating these four MSAs in detail, we found the MUSCLE MSA to be the alignment with the best ratio of number of conserved columns and reasonable compactness. For this reason, we would select the MUSCLE alignment as the basis for further applications.

5.4.2 Discussion

Our evaluation demonstrates that using our visual analysis and comparison approach allows to effectively and efficiently select reasonable MSAs from a set of alternative alignments for further processing. Even though the proposed similarity matrix view in combination with our MSA Shift score measure does not directly allow to assess the quality of the alternative MSAs, it can be used to effectively detect outliers and to get a first overview of the overall alignment differences in the dataset.

The reference MSA visualization based on the Global Residue shift measure can successfully reveal consistently aligned regions across the entire dataset. Likewise, the visualization of the Average Residue Shift measure in the other MSAs allows to retrieve this information with respect to the chosen reference alignment. Both methods thus effectively allow to determine probably correctly aligned regions and to focus on questionable alignment regions for manual refinement.

In our evaluation, however, consistently aligned regions mostly referred to conserved alignment columns. Thus one could argue that measuring the relative rate of conserved or similar amino acids in the alignment columns is sufficient to identify likely correct alignment regions. This would also avoid the computationally expensive MSA comparison. These column-based methods would fail, however, when analyzing structure-based MSAs or alignments of distantly related sequences. In these alignments, conserved columns usually occur less often. In contrast, our approach still allows to identify reliable alignment regions through the detection of consistently aligned regions across alternative MSAs.

Notably, SuiteMSA [Anderson et al. 2011] also provides the possibility to compare MSAs but its capabilities are very limited compared to our approach. For instance, our system represents an all-in-one solution which allows the user to simultaneously compare and analyze multiple large MSAs supported by a fast rendering technique and an unlimited zoom, while in SuiteMSA, only two small MSAs can be effectively compared and accessing other analysis functionalities require the usage of separate and disconnected tools.

Even though our approach provides several benefits for the visual comparison and analysis of MSAs, there are still some limitations. Unlike other visual MSA editors such as Jalview or AliView, our system currently does not provide functions for auto-alignment through external tools or masking mechanisms to hide specific parts of large alignments. Also, we currently do not make use of supporting information such as secondary structure annotations and only provide a limited number of quality measures.

5.5 Conclusion

We presented a novel interactive visual comparison and analysis approach to analyze individual large MSAs and to compare a set of alternative alignments for selecting reliable alignments for further processing. Our system combines automatic comparison and quality measures with different visualizations and highlighting techniques as well as supporting interactions such as an unlimited zoom function. This enables the detection of potentially misaligned regions in individual MSAs. It also allows to reveal differently and consistently aligned areas between alternative MSAs on different levels of detail. For the latter, differences on a global MSA level can be easily and quickly determined using our similarity matrix view showing the N:N pairwise comparison of alternative MSAs. Local alignment differences can be investigated on a per-residue level using our 1:N comparison method and the corresponding visualizations and highlighting techniques.

We evaluated our approach using a set of 8 alternative MSAs based on real biological data (211 HCN ion channel proteins) that is of current research interest in biomedicine. Using our approach, we were able to successfully assess the impact of different MSA algorithms on the resulting alternative alignments. In particular, our approach revealed consistently (and thus probably correct) aligned regions and was able to highlight alignment differences on a per-residue level. This information led to a better understanding of different alignment algorithms and enabled a more reliable selection of a representative alignment for the usage in further applications such as protein structure prediction.

Chapter 6
Conclusion

6.1 Summary

Suboptimal multiple sequence alignments affect various research areas and applications since decades. Most research was focused on the development of more and more complex alignment algorithms to produce better alignment approximations. As a result, a myriad of alignment programs is available to date. Most people are, however, apparently not aware of them and still use outdated tools. But even for these outdated tools, there exists no consensus or at least profound information which of them is most suitable for a given alignment problem. The same holds true for the selection of appropriate scoring models.

Instead of developing yet another complex alignment algorithm that is likely to be overlooked by the biological community, the goal of our work was to significantly improve the MSA results of existing approaches. On one hand, this requires reliable and correct substitution models based on the currently known sequence space that allow existing methods to generate better results. On the other hand, this necessitates tools and methods that allow experts and non-experts to assess the quality of the computed MSAs and to effectively refine them to improve their accuracy.

In order to address the first requirement, we presented two novel substitution models. The CorBLOSUM substitution model [Hess et al. 2016a] presented in Section 4.4 fixes a programming error in the original BLOSUM code. Through the incorrect usage of an integer threshold, sequences are clustered even though they do not fulfill the user-specified minimum similarity value. This error negatively affects the homologous sequence search performance of the original BLOSUM matrices as well as their revised RBLOSUM variants.

Our PFASUM substitution matrices [Keul et al. 2017] presented in Section 4.5 are derived from Pfam seed alignments [Finn et al. 2016] using a novel algorithm. Unlike conventional substitution models, our PFASUM matrices are thus based on manually curated expert ground truth data that reflects the currently known sequence space. Additionally, our PFASUM algorithm incorporates several mechanism to avoid oversampling and handles ambiguous amino acids in a reasonable way.

We fulfilled the second requirement by proposing two different approaches. Our visual analysis and comparison approach [Hess et al. 2016b] presented in Chapter 5 allows to detect reliable and misaligned alignment regions in protein MSAs. It automatically compares alternative MSAs of the same sequence set and visualizes consistently aligned regions and uncertain areas in the MSAs. This in combination with different highlighting modes allows to visually assess the accuracy of MSAs and to determine uncertain regions for further refinement.

Our scientific discovery game Bionigma outsources the cumbersome task of manual MSA refinement to casual players (Section 3.5). It abstracts the protein alignment problem in the form of a puzzle game. The amino acids in the alignment are represented by different game tokens. Their shape, color, and texture visually encode the similarity of the different amino acids. Like one would align beads of identical color in an abacus, the players must align similar tokens to improve their score. Through this, the alignments are successively refined without the need of expert knowledge.

6.2 Discussion

As shown by our exhaustive homology sequence search benchmark presented in Section 4.4.3, there is no longer a reason to use the incorrect BLOSUM62 substitution matrix. Our results showed that our tested CorBLOSUM matrices can significantly outperform their incorrect BLOSUM and RBLOSUM counterparts on modern databases. Notably, our CorBLOSUM model does not claim to be a completely new and innovative substitution model. Instead, it fixes substantial programming errors and thus actually represents the substitution model originally intended by Henikoff and Henikoff [1992]. Although our results demonstrate that the tested CorBLOSUM matrices actually perform better than the incorrect BLOSUM variants, we noticed that the old BLOSUM62 matrices are still commonly used to date. This leaves the impression that the biological community is either not aware of the existence of newer and better methods or simply needs a substantial amount of time to accept new methods and concepts. Similar impressions were also reported by other scientists [Chatzou et al. 2016].

Our PFASUM substitution model showed even better results than the CorBLOSUM matrices. Using our PFASUM31, PFASUM43, and PFASUM60 matrices allows to detect significantly more true homologs than employing widely used substitution matrices. This includes, e.g., the BLOSUM62, PAM250, and VTML200 matrices. Our tests also showed that using PFASUM matrices for the construction of MSAs results in improved accuracy in most cases. Even though we could not evaluate the performance of all substitution matrices available to date, using our PFASUM60 matrix, e.g., is generally a safe choice for biologists that do not have much experience with sequence alignments. We are looking forward to see whether the advantages of these matrices for the computation of sequence alignment will be noticed by the biological community. A few scientists contacted us and reported that they at least plan to test our PFASUM matrices in their tools. Still, we only received profound feedback from a single researcher.

Using our proposed substitution models can lead to significantly more accurate sequence alignments as shown by our benchmark results. Still, the resulting alignments could be further improved through manual inspection of the results and refinement. According to Chatzou et al. [2016], the obvious usefulness of performing these tasks is another aspect that is generally overlooked by the biological community. To our knowledge, most biologists either blindly trust the chosen algorithm and default settings or simply feel not competent enough to inspect the results.

Our proposed visual analysis and comparison approach successfully addresses this problem. Using our system, one can easily determine the accuracy of MSAs without much effort. This is achieved by automatically comparing a set of alternative alignments. Regions that probably can be improved and those that can likely be trusted are comprehensively highlighted, e.g.,

in red or blue. Misalignments can easily be detected through a similarity-based amino acid coloring theme. These features allow even non-experts to quickly assess the accuracy of alignments and to effectively refine them. It also allows to assess the impact of different alignment algorithms and scoring parameters. From this perspective, there are no excuses for not inspecting and improving the quality of MSAs. Additionally, the refinement task can be even successfully outsourced to a crowd of computer gamers as shown by the promising results of our scientific discovery game Bionigma.

In summary, we proposed several approaches that can help to significantly improve the accuracy of sequence alignments without introducing yet another complex alignment algorithm. Notably, our methods enable biologists without profound knowledge in the field of sequence alignments to generate better results without much effort. Nonetheless, our approaches also have their limitations.

Our homologous sequence search and MSA benchmarks do not cover the entire spectrum of substitution matrices that are available to date. Testing all variants of substitution models would have simply required too much time due to the immense computational effort. Our claims of superior matrix performance thus only refer to the comparison of our matrices with a set of widely used substitution matrices.

A notable limitation of our visual analysis and comparison approach is that it lacks some features other visual MSA editors offer. This includes, e.g., options for auto-aligning specific regions or hiding specific parts of the shown alignments. Also, our system currently makes no use of secondary structure information which could be useful for MSA quality assessment.

In view of our scientific discovery game Bionigma, we found that our current alignment scoring model needs further refinement. There is an imbalance between the substitution scores and the applied gap penalties. Still, this problem can be reasonably addressed by applying higher gap penalties. There is also room for improvements regarding Bionigma's game concept. Although most players had a lot of fun playing Bionigma, their feeling of flow and competence should be improved to motivate them in the long run. Notably, we published Bionigma as a beta version and in turn did not yet advertise the game. Hence, we only attracted a relative small number of 64 players so far. Nonetheless, our results demonstrates that our game concept can attract a larger number of players.

6.3 Future work

In the previous section, we identified some limitations of our proposed approaches that could be improved in future work. Beside these limitations, there are additional problems and open research questions that should be addressed in the future. In this section, we discuss these improvements and aspects in detail.

In concordance with previous studies [Price et al. 2005, Hess et al. 2016a], we found that deriving substitution matrices from modern databases covering the currently known sequence space can substantially improve sequence alignments. We are thus planning to derive and evaluate new PFASUM versions for each new Pfam release. Another interesting aspect for future work regarding substitution models are problem-specific substitution matrices. The log-odds scores of substitution matrices inherently depend on the absolute frequency of the amino acid types observed in their originating databases. This raises the interesting question if substitution matrices can deliver better alignment performance, when

they are based on comparable amino acid background frequencies to the sequences that have to be aligned. Additionally, deriving matrices from substitutions between groups of neighboring amino acids instead of pairs of single acids could also be very interesting. These matrices thus would partially grasp the context of a given originating alignment. This could be useful to drive the alignment process into certain directions.

To further improve the game experience of the Bionigma players, we plan to implement additional in-game help mechanics and tutorial improvements. Help mechanics could be, e.g., tools for automatic alignment. The tutorial could be improved by integrating short explanation videos. Other game related enhancements to attract more players are the implementation of achievements, time limited events, and a version for mobile devices. We are also planning to implement additional residue themes to support other substitution models than BLOSUM62 such as our PFASUM matrices. Other improvements regarding the scoring model of Bionigma include further experiments to examine more suitable gap penalties to avoid too many gaps in the players' solutions.

An interesting aspect for future research regarding our visual analysis and comparison approach for MSAs would be a guided MSA analysis. This guide could directly show the user regions of interests and accompanying information. Through this, non-expert users could be further supported in judging the quality of their MSAs. For this purpose, we also want to integrate Bionigma's similarity and bad token highlighting modes. Additionally, we plan to implement missing features compared to other visual MSA editors. This includes interfaces for automatic alignment tools and the integration of secondary structure information.

Beside the future work of our own approaches, there are still several problems affecting the accuracy of sequence alignments that should be addressed in the future. One of the most severe problems in MSA computation and quality analysis is that there is, in our opinion, no sufficient definition of what a correct alignment looks like. The common consensus is that alignments with many conserved columns are probably correct. In contrast, regions containing numerous non-consecutive gaps are typically considered to be unreliable. This common assumption is also reflected by the popular usage of column-based scoring and quality measures such as sum-of-pairs or consensus scores. When analyzing alignments of dissimilar sequences, however, this common consensus simply does not work. Through the overall low sequence similarity in these alignments, most standard measures will probably report almost all columns as "improvable".

Our proposed visual comparison approach circumvents this problem by comparing alternative alignments. Still, measures for the automatic and reliable detection of correct and improvable alignment regions without the need of comparing MSAs would be the better choice. In our opinion, the development of such measures requires, however, rethinking of what makes a good alignment. Instead of rating alignment accuracy by separately assessing the composition of individual alignment columns, the neighboring alignment context should also be taken into account. Are alignment regions that contain numerous mismatches or gaps actually wrong, when they result in almost conserved neighboring regions? Probably not but standard column-only measures cannot grasp this aspect.

Likewise, the role of gaps should be reconsidered. To our impression, there are situations where the introduction of multiple non-consecutive gaps actually makes sense. This includes, e.g., alignment regions that contain very different amino acid types. If these residues could be aligned to form almost conserved columns at the cost of adding multiple smaller

gaps, we would still consider these gaps to be correct. In consequence, new gap scoring models are required that apply penalties depending on the current alignment context. There already exists some adaptive gap penalty models in alignment programs. They are, however, apparently not sufficient to grasp the aforementioned situations.

To our knowledge, one of the biggest problems affecting alignment accuracy is still the biological community itself. As outlined above and as reported by other scientists [Anderson et al. 2011, Chatzou et al. 2016], most people neither make use of improved alignment methods nor inspect their results. Better methods thus cannot have a real impact on alignment accuracy if no one uses them. Therefore, informing the biological community about novel and better alignment methods is crucial to improve the accuracy of sequence alignments in the long run.

We therefore suggest to establish a central alignment service platform that supports experts and non-experts in solving and analyzing their MSAs. This platform could present a central place where developers of alignment algorithms can host and rank their programs using different parameter settings based on state-of-the-art MSA benchmarks. For this purpose, the platform could also provide up-to-date collections of substitution models that could be used in the alignment programs. Based on the rankings, the users could be provided with guidelines which programs are suitable for their alignment task. For example, a questionnaire based on a decision tree reflecting the individual properties of the MSA algorithms could support non-experts in computing reasonable MSAs. Afterwards, this service could also provide tools for analyzing and comparing the computed MSAs. Through this, the users could choose the best alignment for further applications or select regions for refinement. These regions could then be refined by the users themselves or, e.g., with the help of the Bionigma Community.

Appendix A
Supplemental material - Serious games for bioinformatics

No.	Item	GEQ category	GEQ item no.
1	It was aesthetically pleasing	Immersion	14
2	I felt successful	Competence	19
3	I felt bored	Negative affect	18
4	I found it impressive	Immersion	30
5	I lost track of time	Flow	28
6	I felt frustrated	Tension	32
7	I found it tiresome	Negative affect	11
8	I felt irritable	Tension	27
9	I felt skillful	Competence	2
10	I felt completely absorbed	Flow	5
11	I felt content	Positive affect	1
12	I felt challenged	Challenge	29
13	I thought it was hard	Challenge	13
14	I enjoyed it	Positive affect	22

Table A.1: The variation of the in-game experience questionnaire (iGEQ) [IJsselsteijn et al. 2008, Nacke 2009] used for the evaluation of the game experience of our scientific discovery game prototypes and Bionigma. We substituted the iGEQ items one, five, 13, and 14 with the GEQ items 14, 28, 13, and 22, respectively.

No.	Item
1	I found the increase in difficult too drasticy
2	I found the scoring model comprehensible
3	I found the graphical user interface comprehensible
4	I could directly determine the different user interface functions
5	I liked the layout of the graphical user interface
6	I could easily reach all buttons
7	I found the icons comprehensible
8	I could easily distinguish the different colors
9	I found the controls intuitive
10	I could easily learn the controls
11	I found the controls sufficiently precise
12	I could directly see the effect of my inputs
13	I can easily remember the controls
14	I could revert my inputs
15	I could find bugs during play

Table A.2: The second part of the questionnaire used in the user study of our first prototype to assess the different usability aspects of the game.

No.	Item
1	I found the tutorial comprehensive
2	I could understand the meaning of all terms
3	I found the tutorial helpful
4	I could learn all aspects of the game
5	I found the tutorial well-structured
6	I could directly try out the learned content
7	I found the examples helpful
8	I found the texts too long
9	I found the tutorial too long

Table A.3: The third part of the questionnaire used in the user study of our first prototype to assess the quality of the game tutorial.

No.	Item
1	I liked the game
2	I would have liked to play longer
3	I would play the game also in my leisure time
4	I found the tutorial comprehensive
5	I found the tutorial well-structured
6	I found the tutorial helpful
7	I could clearly distinguish the tokens
8	I could clearly identify similar tokens
9	I found the controls intuitive
10	I found the controls sufficiently precise

Table A.4: The second part of the questionnaire used in the user study of Bionigma. It contains items regarding the usability of Bionigma, the usefulness of the tutorial and the provided BLOSUM62-based residue theme, and general game impressions.

Level	SP score			Bionigma score		
	Player	MUSCLE	Ratio	Player	MUSCLE	Ratio
ABC-Transporter-P1b	19338	8438	2.29	8128	6594	1.23
ABC-Transporter2-P1b	35215	14681	2.40	22601	13135	1.72
ABC-Transporter2-P2b	35006	26504	1.32	28466	25376	1.12
Bionigma-B	61523	44182	1.39	53087	43302	1.23
Coronavirus-M-P1b	5279	3891	1.36	4023	3551	1.13
Coronavirus-M-P2b	5547	4408	1.26	4689	4182	1.12
Coronavirus-M-P3b	6075	4308	1.41	4753	4198	1.13
Coronavirus-M-P4b	7032	3810	1.85	4646	3412	1.36
FlxA-like-Protein-P1b	3548	2604	1.36	2610	2258	1.16
Glycoprotein-P1b	12416	7810	1.59	9132	7230	1.26
Hantavirus-G1-P1b	16585	15543	1.07	15891	15361	1.03
Hantavirus-G1-P2b	14609	12261	1.19	11351	11875	0.96
Hepatitis-NS4a-P0b	15536	11150	1.39	13440	10546	1.27
Hepatitis-NS5a_1b-P1b	30024	22389	1.34	26918	21899	1.23
Hepatitis-Virus-P0b	4162	4002	1.04	3950	3878	1.02
Herpes-Glyco-B-P1b	70920	49382	1.44	62308	48570	1.28
Herpes-Glyco-B-P2b	59123	32919	1.80	49453	32185	1.54
Herpes-Glyco-B-P4b	40219	21125	1.90	22131	16677	1.33
Herpes-Glyco-B-P5b	51055	41583	1.23	44903	40295	1.11
Herpes-Glyco-B-P6b	55344	39300	1.41	45618	37734	1.21
Ion-Transporter-P2b	25159	16875	1.49	19495	15555	1.25
Ion-Transporter-P2c	19919	15605	1.28	17975	14525	1.24
Ion-Transporter-P4b	95454	77015	1.24	87612	75365	1.16
Ion-Transporter-P4c	84058	71598	1.17	81812	69948	1.17
NineChanBlues-1	4676	2019	2.32	-3514	-4213	0.83
Peroxiredoxin-P1b	7714	6924	1.11	6900	6646	1.04
Peroxiredoxin-P1c	6755	6639	1.02	6505	6361	1.02
Peroxiredoxin-P2b	21185	17978	1.18	19333	17582	1.10
Peroxiredoxin-P2c	17576	16574	1.06	17180	16186	1.06
Peroxiredoxin-P3b	38422	28846	1.33	33418	27794	1.20
Peroxiredoxin-P3c	31914	27985	1.14	30368	26933	1.13
Usher-Protein-P1b	16985	13718	1.24	15449	13566	1.14
Usher-Protein-P1c	15110	13718	1.10	14674	13566	1.08

Table A.5: The SP and Bionigma scoring results obtained for the different Bionigma levels by the best-performing Bionigma player and the MUSCLE alignment program. The shown ratios represent the factor of score improvement of the player alignments in comparison to the MUSCLE alignments.

Appendix B
Supplemental material - CorBLOSUM substitution matrices

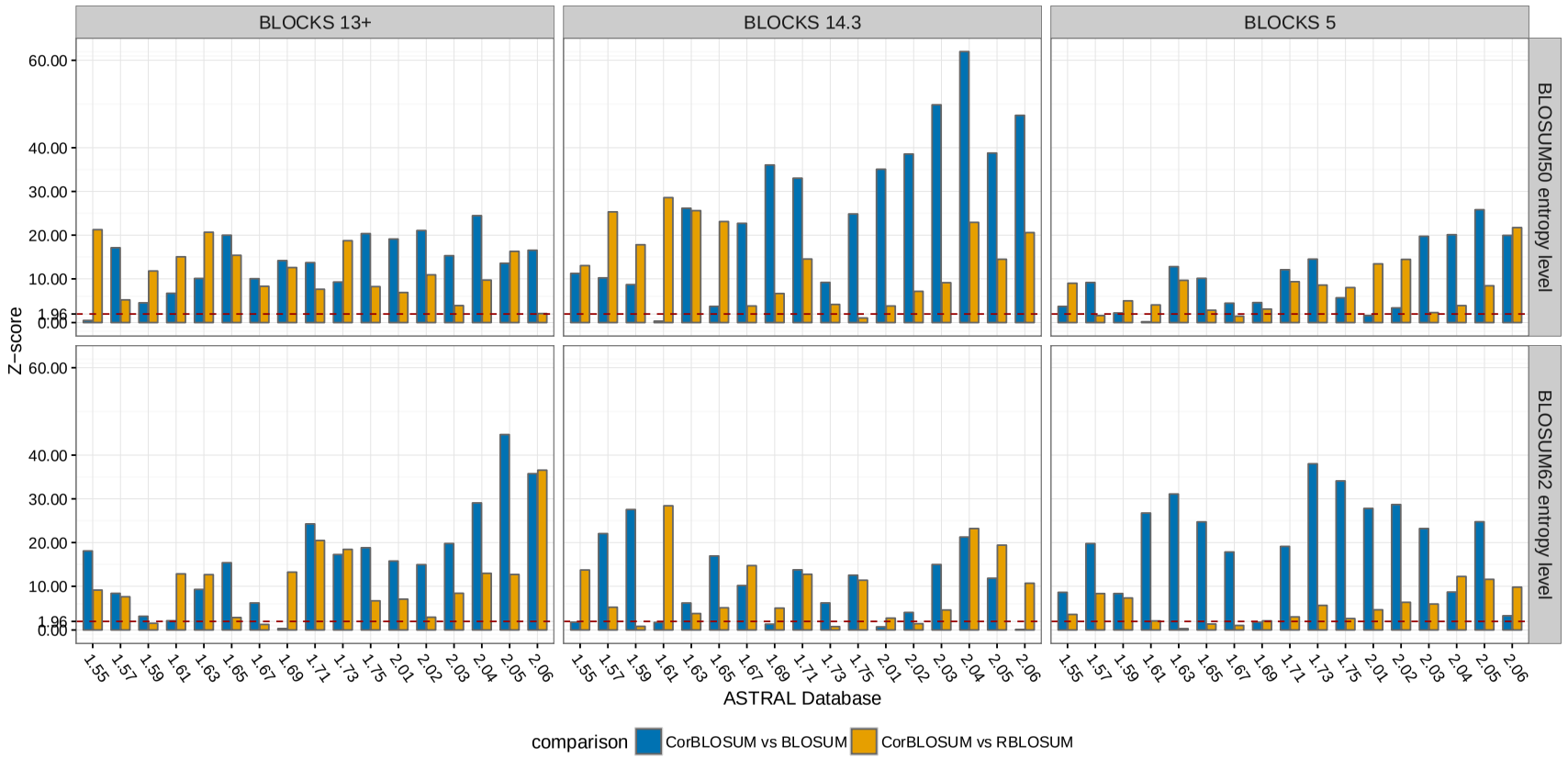


Figure B.1: *Z*-scores for the coverage comparison of CorBLOSUM matrices with BLOSUM and RBLOSUM matrices based on the Concerted Bayesian bootstrapping method for the different versions of the ASTRAL20 subset.

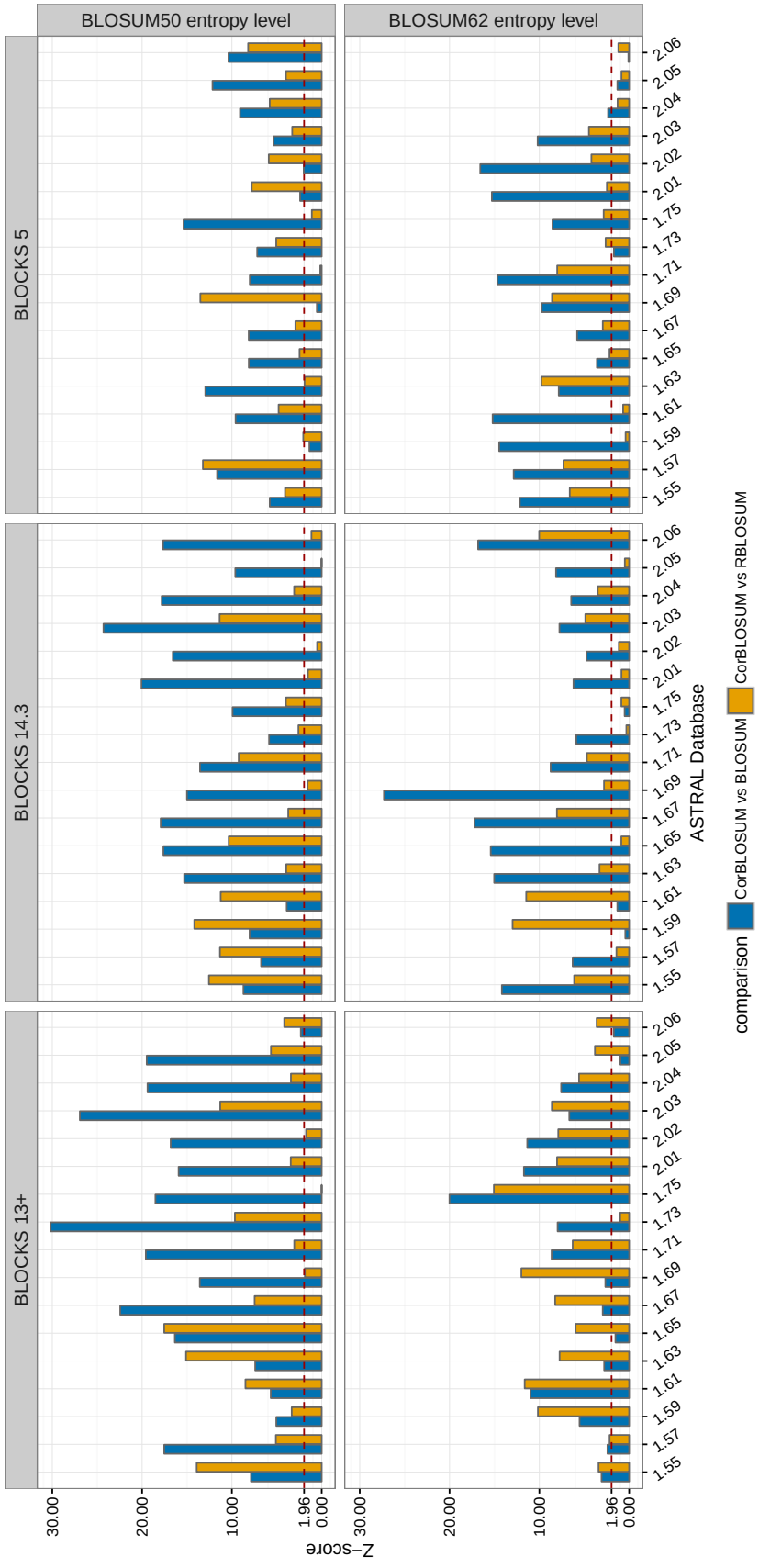


Figure B.2: *Z*-scores for the coverage comparison of CorBLOSUM matrices with BLOSUM and RBLOSUM matrices based on the Concerted Bayesian bootstrapping method for the different versions of the ASTRAL40 subset.

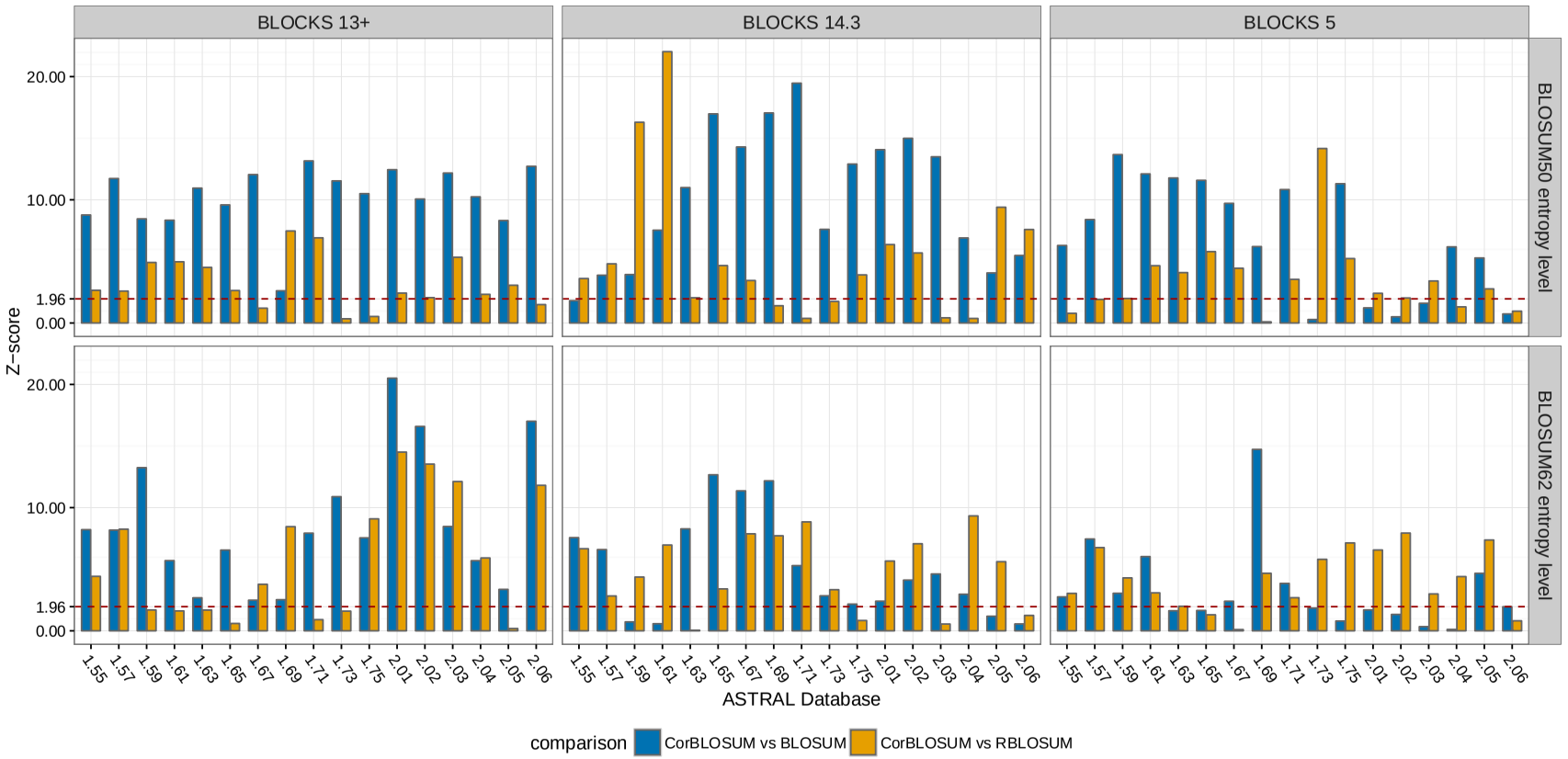


Figure B.3: *Z*-scores for the coverage comparison of CorBLOSUM matrices with BLOSUM and RBLOSUM matrices based on the Concerted Bayesian bootstrapping method for the different versions of the ASTRAL70 subset.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral20	1.55	$BLOSUM50_{5.0}$	14	1	0.1564
Astral20	1.57	$BLOSUM50_{5.0}$	14	1	0.1491
Astral20	1.59	$BLOSUM50_{5.0}$	13	1	0.1423
Astral20	1.61	$BLOSUM50_{5.0}$	13	1	0.1386
Astral20	1.63	$BLOSUM50_{5.0}$	14	1	0.1409
Astral20	1.65	$BLOSUM50_{5.0}$	13	1	0.1454
Astral20	1.67	$BLOSUM50_{5.0}$	12	1	0.1328
Astral20	1.69	$BLOSUM50_{5.0}$	12	1	0.1249
Astral20	1.71	$BLOSUM50_{5.0}$	14	1	0.1243
Astral20	1.73	$BLOSUM50_{5.0}$	12	1	0.1149
Astral20	1.75	$BLOSUM50_{5.0}$	12	1	0.1219
Astral20	2.01	$BLOSUM50_{5.0}$	12	1	0.1207
Astral20	2.02	$BLOSUM50_{5.0}$	12	1	0.1214
Astral20	2.03	$BLOSUM50_{5.0}$	13	1	0.1190
Astral20	2.04	$BLOSUM50_{5.0}$	15	1	0.1333
Astral20	2.05	$BLOSUM50_{5.0}$	15	1	0.1404
Astral20	2.06	$BLOSUM50_{5.0}$	15	1	0.1474
Astral20	1.55	$BLOSUM62_{5.0}$	9	1	0.1518
Astral20	1.57	$BLOSUM62_{5.0}$	6	2	0.1466
Astral20	1.59	$BLOSUM62_{5.0}$	9	1	0.1342
Astral20	1.61	$BLOSUM62_{5.0}$	9	1	0.1335
Astral20	1.63	$BLOSUM62_{5.0}$	9	1	0.1391
Astral20	1.65	$BLOSUM62_{5.0}$	9	1	0.1390
Astral20	1.67	$BLOSUM62_{5.0}$	11	1	0.1321
Astral20	1.69	$BLOSUM62_{5.0}$	9	1	0.1225
Astral20	1.71	$BLOSUM62_{5.0}$	9	1	0.1257
Astral20	1.73	$BLOSUM62_{5.0}$	9	1	0.1163
Astral20	1.75	$BLOSUM62_{5.0}$	9	1	0.1190
Astral20	2.01	$BLOSUM62_{5.0}$	8	1	0.1193
Astral20	2.02	$BLOSUM62_{5.0}$	8	1	0.1195
Astral20	2.03	$BLOSUM62_{5.0}$	8	1	0.1207
Astral20	2.04	$BLOSUM62_{5.0}$	10	1	0.1321
Astral20	2.05	$BLOSUM62_{5.0}$	10	1	0.1366
Astral20	2.06	$BLOSUM62_{5.0}$	10	1	0.1465
Astral20	1.55	$CorBLOSUM49_{5.0}$	14	1	0.1552
Astral20	1.57	$CorBLOSUM49_{5.0}$	11	2	0.1514
Astral20	1.59	$CorBLOSUM49_{5.0}$	12	1	0.1426
Astral20	1.61	$CorBLOSUM49_{5.0}$	13	1	0.1383
Astral20	1.63	$CorBLOSUM49_{5.0}$	16	1	0.1439
Astral20	1.65	$CorBLOSUM49_{5.0}$	16	1	0.1429
Astral20	1.67	$CorBLOSUM49_{5.0}$	14	1	0.1319
Astral20	1.69	$CorBLOSUM49_{5.0}$	17	1	0.1257
Astral20	1.71	$CorBLOSUM49_{5.0}$	15	1	0.1268
Astral20	1.73	$CorBLOSUM49_{5.0}$	16	1	0.1173
Astral20	1.75	$CorBLOSUM49_{5.0}$	15	1	0.1210
Astral20	2.01	$CorBLOSUM49_{5.0}$	12	1	0.1209
Astral20	2.02	$CorBLOSUM49_{5.0}$	12	1	0.1209
Astral20	2.03	$CorBLOSUM49_{5.0}$	15	1	0.1219
Astral20	2.04	$CorBLOSUM49_{5.0}$	14	1	0.1365
Astral20	2.05	$CorBLOSUM49_{5.0}$	14	1	0.1447
Astral20	2.06	$CorBLOSUM49_{5.0}$	16	1	0.1505

Subset	Version	Matrix	Gop	Ext	Cov.
Astral20	1.55	$CorBLOSUM61_{5.0}$	9	1	0.1495
Astral20	1.57	$CorBLOSUM61_{5.0}$	9	1	0.1421
Astral20	1.59	$CorBLOSUM61_{5.0}$	8	1	0.1324
Astral20	1.61	$CorBLOSUM61_{5.0}$	8	1	0.1275
Astral20	1.63	$CorBLOSUM61_{5.0}$	8	1	0.1321
Astral20	1.65	$CorBLOSUM61_{5.0}$	11	1	0.1338
Astral20	1.67	$CorBLOSUM61_{5.0}$	9	1	0.1288
Astral20	1.69	$CorBLOSUM61_{5.0}$	9	1	0.1220
Astral20	1.71	$CorBLOSUM61_{5.0}$	9	1	0.1224
Astral20	1.73	$CorBLOSUM61_{5.0}$	8	1	0.1103
Astral20	1.75	$CorBLOSUM61_{5.0}$	8	1	0.1140
Astral20	2.01	$CorBLOSUM61_{5.0}$	8	1	0.1151
Astral20	2.02	$CorBLOSUM61_{5.0}$	8	1	0.1151
Astral20	2.03	$CorBLOSUM61_{5.0}$	9	1	0.1165
Astral20	2.04	$CorBLOSUM61_{5.0}$	9	1	0.1338
Astral20	2.05	$CorBLOSUM61_{5.0}$	9	1	0.1405
Astral20	2.06	$CorBLOSUM61_{5.0}$	9	1	0.1473
Astral20	1.55	$RBLOSUM52_{5.0}$	14	1	0.1577
Astral20	1.57	$RBLOSUM52_{5.0}$	14	1	0.1520
Astral20	1.59	$RBLOSUM52_{5.0}$	14	1	0.1416
Astral20	1.61	$RBLOSUM52_{5.0}$	12	1	0.1372
Astral20	1.63	$RBLOSUM52_{5.0}$	15	1	0.1419
Astral20	1.65	$RBLOSUM52_{5.0}$	16	1	0.1425
Astral20	1.67	$RBLOSUM52_{5.0}$	12	1	0.1318
Astral20	1.69	$RBLOSUM52_{5.0}$	17	1	0.1253
Astral20	1.71	$RBLOSUM52_{5.0}$	15	1	0.1248
Astral20	1.73	$RBLOSUM52_{5.0}$	15	1	0.1160
Astral20	1.75	$RBLOSUM52_{5.0}$	14	1	0.1198
Astral20	2.01	$RBLOSUM52_{5.0}$	12	1	0.1184
Astral20	2.02	$RBLOSUM52_{5.0}$	16	1	0.1186
Astral20	2.03	$RBLOSUM52_{5.0}$	17	1	0.1216
Astral20	2.04	$RBLOSUM52_{5.0}$	15	1	0.1361
Astral20	2.05	$RBLOSUM52_{5.0}$	15	1	0.1432
Astral20	2.06	$RBLOSUM52_{5.0}$	16	1	0.1544
Astral20	1.55	$RBLOSUM64_{5.0}$	9	1	0.1504
Astral20	1.57	$RBLOSUM64_{5.0}$	9	1	0.1405
Astral20	1.59	$RBLOSUM64_{5.0}$	8	1	0.1310
Astral20	1.61	$RBLOSUM64_{5.0}$	8	1	0.1269
Astral20	1.63	$RBLOSUM64_{5.0}$	8	1	0.1321
Astral20	1.65	$RBLOSUM64_{5.0}$	8	1	0.1336
Astral20	1.67	$RBLOSUM64_{5.0}$	8	1	0.1288
Astral20	1.69	$RBLOSUM64_{5.0}$	9	1	0.1225
Astral20	1.71	$RBLOSUM64_{5.0}$	9	1	0.1218
Astral20	1.73	$RBLOSUM64_{5.0}$	7	1	0.1112
Astral20	1.75	$RBLOSUM64_{5.0}$	8	1	0.1137
Astral20	2.01	$RBLOSUM64_{5.0}$	8	1	0.1140
Astral20	2.02	$RBLOSUM64_{5.0}$	8	1	0.1137
Astral20	2.03	$RBLOSUM64_{5.0}$	9	1	0.1158
Astral20	2.04	$RBLOSUM64_{5.0}$	9	1	0.1317
Astral20	2.05	$RBLOSUM64_{5.0}$	9	1	0.1386
Astral20	2.06	$RBLOSUM64_{5.0}$	9	1	0.1456

Table B.1: Highest achieved coverage values (Cov.) on the different ASTRAL20 subset versions for the six BLOCKS 5.0-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral20	1.55	BLOSUM50$_{13+}$	14	2	0.1487
Astral20	1.57	BLOSUM50$_{13+}$	16	2	0.1413
Astral20	1.59	BLOSUM50$_{13+}$	15	1	0.1335
Astral20	1.61	BLOSUM50$_{13+}$	15	1	0.1302
Astral20	1.63	BLOSUM50$_{13+}$	16	1	0.1335
Astral20	1.65	BLOSUM50$_{13+}$	15	1	0.1279
Astral20	1.67	BLOSUM50$_{13+}$	17	1	0.1308
Astral20	1.69	BLOSUM50$_{13+}$	19	1	0.1230
Astral20	1.71	BLOSUM50$_{13+}$	18	1	0.1217
Astral20	1.73	BLOSUM50$_{13+}$	18	1	0.1157
Astral20	1.75	BLOSUM50$_{13+}$	17	1	0.1153
Astral20	2.01	BLOSUM50$_{13+}$	19	1	0.1170
Astral20	2.02	BLOSUM50$_{13+}$	19	1	0.1169
Astral20	2.03	BLOSUM50$_{13+}$	19	1	0.1188
Astral20	2.04	BLOSUM50$_{13+}$	18	1	0.1330
Astral20	2.05	BLOSUM50$_{13+}$	18	1	0.1379
Astral20	2.06	BLOSUM50$_{13+}$	16	2	0.1413
Astral20	1.55	BLOSUM62$_{13+}$	12	2	0.1560
Astral20	1.57	BLOSUM62$_{13+}$	13	1	0.1515
Astral20	1.59	BLOSUM62$_{13+}$	14	1	0.1448
Astral20	1.61	BLOSUM62$_{13+}$	12	1	0.1402
Astral20	1.63	BLOSUM62$_{13+}$	14	1	0.1434
Astral20	1.65	BLOSUM62$_{13+}$	13	1	0.1399
Astral20	1.67	BLOSUM62$_{13+}$	14	1	0.1377
Astral20	1.69	BLOSUM62$_{13+}$	14	1	0.1294
Astral20	1.71	BLOSUM62$_{13+}$	14	1	0.1302
Astral20	1.73	BLOSUM62$_{13+}$	14	1	0.1225
Astral20	1.75	BLOSUM62$_{13+}$	14	1	0.1231
Astral20	2.01	BLOSUM62$_{13+}$	14	1	0.1245
Astral20	2.02	BLOSUM62$_{13+}$	14	1	0.1246
Astral20	2.03	BLOSUM62$_{13+}$	15	1	0.1241
Astral20	2.04	BLOSUM62$_{13+}$	15	1	0.1373
Astral20	2.05	BLOSUM62$_{13+}$	14	1	0.1416
Astral20	2.06	BLOSUM62$_{13+}$	15	1	0.1466
Astral20	1.55	CorBLOSUM57$_{13+}$	16	1	0.1489
Astral20	1.57	CorBLOSUM57$_{13+}$	13	2	0.1452
Astral20	1.59	CorBLOSUM57$_{13+}$	16	1	0.1347
Astral20	1.61	CorBLOSUM57$_{13+}$	16	1	0.1314
Astral20	1.63	CorBLOSUM57$_{13+}$	18	1	0.1313
Astral20	1.65	CorBLOSUM57$_{13+}$	18	1	0.1322
Astral20	1.67	CorBLOSUM57$_{13+}$	17	2	0.1328
Astral20	1.69	CorBLOSUM57$_{13+}$	18	1	0.1256
Astral20	1.71	CorBLOSUM57$_{13+}$	18	1	0.1240
Astral20	1.73	CorBLOSUM57$_{13+}$	20	1	0.1173
Astral20	1.75	CorBLOSUM57$_{13+}$	20	1	0.1184
Astral20	2.01	CorBLOSUM57$_{13+}$	20	1	0.1202
Astral20	2.02	CorBLOSUM57$_{13+}$	20	1	0.1202
Astral20	2.03	CorBLOSUM57$_{13+}$	18	1	0.1211
Astral20	2.04	CorBLOSUM57$_{13+}$	13	2	0.1372
Astral20	2.05	CorBLOSUM57$_{13+}$	13	2	0.1402
Astral20	2.06	CorBLOSUM57$_{13+}$	18	1	0.1438

Subset	Version	Matrix	Gop	Ext	Cov.
Astral20	1.55	CorBLOSUM66$_{13+}$	11	2	0.1611
Astral20	1.57	CorBLOSUM66$_{13+}$	12	1	0.1491
Astral20	1.59	CorBLOSUM66$_{13+}$	13	1	0.1439
Astral20	1.61	CorBLOSUM66$_{13+}$	12	1	0.1395
Astral20	1.63	CorBLOSUM66$_{13+}$	13	1	0.1416
Astral20	1.65	CorBLOSUM66$_{13+}$	13	1	0.1431
Astral20	1.67	CorBLOSUM66$_{13+}$	14	1	0.1387
Astral20	1.69	CorBLOSUM66$_{13+}$	14	1	0.1293
Astral20	1.71	CorBLOSUM66$_{13+}$	14	1	0.1255
Astral20	1.73	CorBLOSUM66$_{13+}$	15	1	0.1195
Astral20	1.75	CorBLOSUM66$_{13+}$	14	1	0.1202
Astral20	2.01	CorBLOSUM66$_{13+}$	14	1	0.1272
Astral20	2.02	CorBLOSUM66$_{13+}$	14	1	0.1272
Astral20	2.03	CorBLOSUM66$_{13+}$	14	1	0.1276
Astral20	2.04	CorBLOSUM66$_{13+}$	14	1	0.1421
Astral20	2.05	CorBLOSUM66$_{13+}$	14	1	0.1494
Astral20	2.06	CorBLOSUM66$_{13+}$	15	1	0.1528
Astral20	1.55	RBLOSUM59$_{13+}$	16	2	0.1547
Astral20	1.57	RBLOSUM59$_{13+}$	16	1	0.1467
Astral20	1.59	RBLOSUM59$_{13+}$	15	1	0.1375
Astral20	1.61	RBLOSUM59$_{13+}$	15	1	0.1346
Astral20	1.63	RBLOSUM59$_{13+}$	15	1	0.1360
Astral20	1.65	RBLOSUM59$_{13+}$	17	1	0.1360
Astral20	1.67	RBLOSUM59$_{13+}$	18	1	0.1344
Astral20	1.69	RBLOSUM59$_{13+}$	18	1	0.1280
Astral20	1.71	RBLOSUM59$_{13+}$	18	1	0.1254
Astral20	1.73	RBLOSUM59$_{13+}$	18	1	0.1206
Astral20	1.75	RBLOSUM59$_{13+}$	20	1	0.1199
Astral20	2.01	RBLOSUM59$_{13+}$	20	1	0.1213
Astral20	2.02	RBLOSUM59$_{13+}$	20	1	0.1219
Astral20	2.03	RBLOSUM59$_{13+}$	19	1	0.1206
Astral20	2.04	RBLOSUM59$_{13+}$	14	2	0.1356
Astral20	2.05	RBLOSUM59$_{13+}$	14	2	0.1374
Astral20	2.06	RBLOSUM59$_{13+}$	16	2	0.1432
Astral20	1.55	RBLOSUM69$_{13+}$	12	2	0.1634
Astral20	1.57	RBLOSUM69$_{13+}$	13	1	0.1512
Astral20	1.59	RBLOSUM69$_{13+}$	12	1	0.1444
Astral20	1.61	RBLOSUM69$_{13+}$	13	1	0.1425
Astral20	1.63	RBLOSUM69$_{13+}$	13	1	0.1444
Astral20	1.65	RBLOSUM69$_{13+}$	13	1	0.1437
Astral20	1.67	RBLOSUM69$_{13+}$	13	1	0.1392
Astral20	1.69	RBLOSUM69$_{13+}$	13	1	0.1318
Astral20	1.71	RBLOSUM69$_{13+}$	15	1	0.1294
Astral20	1.73	RBLOSUM69$_{13+}$	14	1	0.1227
Astral20	1.75	RBLOSUM69$_{13+}$	15	1	0.1213
Astral20	2.01	RBLOSUM69$_{13+}$	14	1	0.1262
Astral20	2.02	RBLOSUM69$_{13+}$	14	1	0.1265
Astral20	2.03	RBLOSUM69$_{13+}$	14	1	0.1262
Astral20	2.04	RBLOSUM69$_{13+}$	14	1	0.1400
Astral20	2.05	RBLOSUM69$_{13+}$	13	1	0.1471
Astral20	2.06	RBLOSUM69$_{13+}$	13	1	0.1465

Table B.2: Highest achieved coverage values (Cov.) on the different ASTRAL20 subset versions for the six BLOCKS 13+-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral20	1.55	$BLOSUM50_{14.3}$	18	1	0.1392
Astral20	1.57	$BLOSUM50_{14.3}$	19	1	0.1377
Astral20	1.59	$BLOSUM50_{14.3}$	19	1	0.1294
Astral20	1.61	$BLOSUM50_{14.3}$	16	1	0.1273
Astral20	1.63	$BLOSUM50_{14.3}$	18	1	0.1292
Astral20	1.65	$BLOSUM50_{14.3}$	20	1	0.1303
Astral20	1.67	$BLOSUM50_{14.3}$	20	1	0.1281
Astral20	1.69	$BLOSUM50_{14.3}$	17	2	0.1174
Astral20	1.71	$BLOSUM50_{14.3}$	17	2	0.1179
Astral20	1.73	$BLOSUM50_{14.3}$	18	2	0.1142
Astral20	1.75	$BLOSUM50_{14.3}$	19	2	0.1169
Astral20	2.01	$BLOSUM50_{14.3}$	19	2	0.1155
Astral20	2.02	$BLOSUM50_{14.3}$	19	2	0.1158
Astral20	2.03	$BLOSUM50_{14.3}$	17	2	0.1157
Astral20	2.04	$BLOSUM50_{14.3}$	20	2	0.1300
Astral20	2.05	$BLOSUM50_{14.3}$	18	2	0.1363
Astral20	2.06	$BLOSUM50_{14.3}$	19	2	0.1367
Astral20	1.55	$BLOSUM62_{14.3}$	15	2	0.1518
Astral20	1.57	$BLOSUM62_{14.3}$	19	1	0.1412
Astral20	1.59	$BLOSUM62_{14.3}$	15	1	0.1334
Astral20	1.61	$BLOSUM62_{14.3}$	17	1	0.1310
Astral20	1.63	$BLOSUM62_{14.3}$	19	1	0.1326
Astral20	1.65	$BLOSUM62_{14.3}$	18	1	0.1366
Astral20	1.67	$BLOSUM62_{14.3}$	16	2	0.1350
Astral20	1.69	$BLOSUM62_{14.3}$	15	2	0.1276
Astral20	1.71	$BLOSUM62_{14.3}$	16	2	0.1278
Astral20	1.73	$BLOSUM62_{14.3}$	20	1	0.1210
Astral20	1.75	$BLOSUM62_{14.3}$	16	2	0.1245
Astral20	2.01	$BLOSUM62_{14.3}$	16	2	0.1242
Astral20	2.02	$BLOSUM62_{14.3}$	16	2	0.1246
Astral20	2.03	$BLOSUM62_{14.3}$	20	1	0.1257
Astral20	2.04	$BLOSUM62_{14.3}$	16	2	0.1412
Astral20	2.05	$BLOSUM62_{14.3}$	16	2	0.1471
Astral20	2.06	$BLOSUM62_{14.3}$	15	2	0.1463
Astral20	1.55	$CorBLOSUM57_{14.3}$	17	1	0.1421
Astral20	1.57	$CorBLOSUM57_{14.3}$	18	1	0.1353
Astral20	1.59	$CorBLOSUM57_{14.3}$	17	1	0.1272
Astral20	1.61	$CorBLOSUM57_{14.3}$	16	1	0.1275
Astral20	1.63	$CorBLOSUM57_{14.3}$	16	1	0.1239
Astral20	1.65	$CorBLOSUM57_{14.3}$	19	1	0.1296
Astral20	1.67	$CorBLOSUM57_{14.3}$	20	1	0.1325
Astral20	1.69	$CorBLOSUM57_{14.3}$	20	1	0.1241
Astral20	1.71	$CorBLOSUM57_{14.3}$	20	1	0.1240
Astral20	1.73	$CorBLOSUM57_{14.3}$	19	2	0.1157
Astral20	1.75	$CorBLOSUM57_{14.3}$	19	2	0.1209
Astral20	2.01	$CorBLOSUM57_{14.3}$	17	2	0.1211
Astral20	2.02	$CorBLOSUM57_{14.3}$	17	2	0.1215
Astral20	2.03	$CorBLOSUM57_{14.3}$	17	2	0.1238
Astral20	2.04	$CorBLOSUM57_{14.3}$	17	2	0.1403
Astral20	2.05	$CorBLOSUM57_{14.3}$	17	2	0.1431
Astral20	2.06	$CorBLOSUM57_{14.3}$	19	2	0.1449

Subset	Version	Matrix	Gop	Ext	Cov.
Astral20	1.55	$CorBLOSUM67_{14.3}$	14	2	0.1521
Astral20	1.57	$CorBLOSUM67_{14.3}$	15	1	0.1460
Astral20	1.59	$CorBLOSUM67_{14.3}$	15	1	0.1395
Astral20	1.61	$CorBLOSUM67_{14.3}$	15	1	0.1313
Astral20	1.63	$CorBLOSUM67_{14.3}$	17	1	0.1336
Astral20	1.65	$CorBLOSUM67_{14.3}$	18	1	0.1402
Astral20	1.67	$CorBLOSUM67_{14.3}$	18	1	0.1370
Astral20	1.69	$CorBLOSUM67_{14.3}$	18	1	0.1275
Astral20	1.71	$CorBLOSUM67_{14.3}$	18	1	0.1300
Astral20	1.73	$CorBLOSUM67_{14.3}$	18	1	0.1223
Astral20	1.75	$CorBLOSUM67_{14.3}$	19	1	0.1223
Astral20	2.01	$CorBLOSUM67_{14.3}$	18	1	0.1241
Astral20	2.02	$CorBLOSUM67_{14.3}$	18	1	0.1241
Astral20	2.03	$CorBLOSUM67_{14.3}$	18	1	0.1281
Astral20	2.04	$CorBLOSUM67_{14.3}$	15	2	0.1380
Astral20	2.05	$CorBLOSUM67_{14.3}$	14	2	0.1451
Astral20	2.06	$CorBLOSUM67_{14.3}$	15	2	0.1464
Astral20	1.55	$RBLOSUM59_{14.3}$	17	1	0.1456
Astral20	1.57	$RBLOSUM59_{14.3}$	16	1	0.1416
Astral20	1.59	$RBLOSUM59_{14.3}$	17	1	0.1310
Astral20	1.61	$RBLOSUM59_{14.3}$	16	1	0.1335
Astral20	1.63	$RBLOSUM59_{14.3}$	17	1	0.1292
Astral20	1.65	$RBLOSUM59_{14.3}$	18	1	0.1345
Astral20	1.67	$RBLOSUM59_{14.3}$	20	1	0.1333
Astral20	1.69	$RBLOSUM59_{14.3}$	19	1	0.1227
Astral20	1.71	$RBLOSUM59_{14.3}$	19	1	0.1215
Astral20	1.73	$RBLOSUM59_{14.3}$	20	1	0.1165
Astral20	1.75	$RBLOSUM59_{14.3}$	20	1	0.1211
Astral20	2.01	$RBLOSUM59_{14.3}$	20	1	0.1205
Astral20	2.02	$RBLOSUM59_{14.3}$	20	1	0.1207
Astral20	2.03	$RBLOSUM59_{14.3}$	20	1	0.1222
Astral20	2.04	$RBLOSUM59_{14.3}$	16	2	0.1366
Astral20	2.05	$RBLOSUM59_{14.3}$	16	2	0.1407
Astral20	2.06	$RBLOSUM59_{14.3}$	18	2	0.1415
Astral20	1.55	$RBLOSUM69_{14.3}$	16	1	0.1484
Astral20	1.57	$RBLOSUM69_{14.3}$	19	1	0.1479
Astral20	1.59	$RBLOSUM69_{14.3}$	18	1	0.1400
Astral20	1.61	$RBLOSUM69_{14.3}$	18	1	0.1377
Astral20	1.63	$RBLOSUM69_{14.3}$	18	1	0.1345
Astral20	1.65	$RBLOSUM69_{14.3}$	20	1	0.1390
Astral20	1.67	$RBLOSUM69_{14.3}$	19	1	0.1340
Astral20	1.69	$RBLOSUM69_{14.3}$	19	1	0.1266
Astral20	1.71	$RBLOSUM69_{14.3}$	20	1	0.1276
Astral20	1.73	$RBLOSUM69_{14.3}$	19	1	0.1220
Astral20	1.75	$RBLOSUM69_{14.3}$	19	1	0.1207
Astral20	2.01	$RBLOSUM69_{14.3}$	18	1	0.1238
Astral20	2.02	$RBLOSUM69_{14.3}$	18	1	0.1238
Astral20	2.03	$RBLOSUM69_{14.3}$	18	1	0.1273
Astral20	2.04	$RBLOSUM69_{14.3}$	20	1	0.1340
Astral20	2.05	$RBLOSUM69_{14.3}$	20	1	0.1418
Astral20	2.06	$RBLOSUM69_{14.3}$	20	1	0.1445

Table B.3: Highest achieved coverage values (Cov.) on the different ASTRAL20 subset versions for the six BLOCKS 14.3-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral40	1.55	$BLOSUM50_{5.0}$	13	1	0.3836
Astral40	1.57	$BLOSUM50_{5.0}$	13	1	0.3765
Astral40	1.59	$BLOSUM50_{5.0}$	13	1	0.3741
Astral40	1.61	$BLOSUM50_{5.0}$	14	1	0.3744
Astral40	1.63	$BLOSUM50_{5.0}$	13	1	0.3779
Astral40	1.65	$BLOSUM50_{5.0}$	14	1	0.3764
Astral40	1.67	$BLOSUM50_{5.0}$	13	1	0.3838
Astral40	1.69	$BLOSUM50_{5.0}$	15	1	0.3715
Astral40	1.71	$BLOSUM50_{5.0}$	15	1	0.3681
Astral40	1.73	$BLOSUM50_{5.0}$	15	1	0.3710
Astral40	1.75	$BLOSUM50_{5.0}$	12	2	0.3730
Astral40	2.01	$BLOSUM50_{5.0}$	12	2	0.3992
Astral40	2.02	$BLOSUM50_{5.0}$	12	2	0.3992
Astral40	2.03	$BLOSUM50_{5.0}$	11	2	0.4128
Astral40	2.04	$BLOSUM50_{5.0}$	12	1	0.4236
Astral40	2.05	$BLOSUM50_{5.0}$	11	1	0.4268
Astral40	2.06	$BLOSUM50_{5.0}$	11	2	0.4371
Astral40	1.55	$BLOSUM62_{5.0}$	9	1	0.3789
Astral40	1.57	$BLOSUM62_{5.0}$	6	2	0.3730
Astral40	1.59	$BLOSUM62_{5.0}$	9	1	0.3732
Astral40	1.61	$BLOSUM62_{5.0}$	9	1	0.3757
Astral40	1.63	$BLOSUM62_{5.0}$	9	1	0.3790
Astral40	1.65	$BLOSUM62_{5.0}$	8	1	0.3765
Astral40	1.67	$BLOSUM62_{5.0}$	10	1	0.3834
Astral40	1.69	$BLOSUM62_{5.0}$	10	1	0.3694
Astral40	1.71	$BLOSUM62_{5.0}$	10	1	0.3653
Astral40	1.73	$BLOSUM62_{5.0}$	9	1	0.3675
Astral40	1.75	$BLOSUM62_{5.0}$	10	1	0.3731
Astral40	2.01	$BLOSUM62_{5.0}$	10	1	0.3977
Astral40	2.02	$BLOSUM62_{5.0}$	10	1	0.3977
Astral40	2.03	$BLOSUM62_{5.0}$	9	1	0.4112
Astral40	2.04	$BLOSUM62_{5.0}$	10	1	0.4224
Astral40	2.05	$BLOSUM62_{5.0}$	10	1	0.4245
Astral40	2.06	$BLOSUM62_{5.0}$	10	1	0.4346
Astral40	1.55	$CorBLOSUM49_{5.0}$	11	2	0.3822
Astral40	1.57	$CorBLOSUM49_{5.0}$	13	1	0.3734
Astral40	1.59	$CorBLOSUM49_{5.0}$	14	1	0.3750
Astral40	1.61	$CorBLOSUM49_{5.0}$	14	1	0.3770
Astral40	1.63	$CorBLOSUM49_{5.0}$	13	1	0.3815
Astral40	1.65	$CorBLOSUM49_{5.0}$	14	1	0.3786
Astral40	1.67	$CorBLOSUM49_{5.0}$	14	1	0.3855
Astral40	1.69	$CorBLOSUM49_{5.0}$	15	1	0.3718
Astral40	1.71	$CorBLOSUM49_{5.0}$	16	1	0.3699
Astral40	1.73	$CorBLOSUM49_{5.0}$	15	1	0.3726
Astral40	1.75	$CorBLOSUM49_{5.0}$	15	1	0.3760
Astral40	2.01	$CorBLOSUM49_{5.0}$	15	1	0.3990
Astral40	2.02	$CorBLOSUM49_{5.0}$	15	1	0.3990
Astral40	2.03	$CorBLOSUM49_{5.0}$	15	1	0.4141
Astral40	2.04	$CorBLOSUM49_{5.0}$	15	1	0.4253
Astral40	2.05	$CorBLOSUM49_{5.0}$	14	1	0.4288
Astral40	2.06	$CorBLOSUM49_{5.0}$	15	1	0.4389

Subset	Version	Matrix	Gop	Ext	Cov.
Astral40	1.55	$CorBLOSUM61_{5.0}$	8	1	0.3755
Astral40	1.57	$CorBLOSUM61_{5.0}$	8	1	0.3693
Astral40	1.59	$CorBLOSUM61_{5.0}$	8	1	0.3689
Astral40	1.61	$CorBLOSUM61_{5.0}$	8	1	0.3713
Astral40	1.63	$CorBLOSUM61_{5.0}$	8	1	0.3770
Astral40	1.65	$CorBLOSUM61_{5.0}$	9	1	0.3757
Astral40	1.67	$CorBLOSUM61_{5.0}$	8	1	0.3821
Astral40	1.69	$CorBLOSUM61_{5.0}$	9	1	0.3668
Astral40	1.71	$CorBLOSUM61_{5.0}$	9	1	0.3626
Astral40	1.73	$CorBLOSUM61_{5.0}$	9	1	0.3670
Astral40	1.75	$CorBLOSUM61_{5.0}$	9	1	0.3716
Astral40	2.01	$CorBLOSUM61_{5.0}$	9	1	0.3945
Astral40	2.02	$CorBLOSUM61_{5.0}$	9	1	0.3945
Astral40	2.03	$CorBLOSUM61_{5.0}$	9	1	0.4091
Astral40	2.04	$CorBLOSUM61_{5.0}$	9	1	0.4230
Astral40	2.05	$CorBLOSUM61_{5.0}$	9	1	0.4247
Astral40	2.06	$CorBLOSUM61_{5.0}$	9	1	0.4346
Astral40	1.55	$RBLOSUM52_{5.0}$	11	2	0.3834
Astral40	1.57	$RBLOSUM52_{5.0}$	11	2	0.3768
Astral40	1.59	$RBLOSUM52_{5.0}$	13	1	0.3738
Astral40	1.61	$RBLOSUM52_{5.0}$	13	1	0.3760
Astral40	1.63	$RBLOSUM52_{5.0}$	14	1	0.3820
Astral40	1.65	$RBLOSUM52_{5.0}$	14	1	0.3791
Astral40	1.67	$RBLOSUM52_{5.0}$	14	1	0.3852
Astral40	1.69	$RBLOSUM52_{5.0}$	15	1	0.3745
Astral40	1.71	$RBLOSUM52_{5.0}$	16	1	0.3696
Astral40	1.73	$RBLOSUM52_{5.0}$	15	1	0.3735
Astral40	1.75	$RBLOSUM52_{5.0}$	15	1	0.3760
Astral40	2.01	$RBLOSUM52_{5.0}$	15	1	0.4001
Astral40	2.02	$RBLOSUM52_{5.0}$	15	1	0.4001
Astral40	2.03	$RBLOSUM52_{5.0}$	15	1	0.4146
Astral40	2.04	$RBLOSUM52_{5.0}$	14	1	0.4262
Astral40	2.05	$RBLOSUM52_{5.0}$	14	1	0.4298
Astral40	2.06	$RBLOSUM52_{5.0}$	15	1	0.4375
Astral40	1.55	$RBLOSUM64_{5.0}$	8	1	0.3770
Astral40	1.57	$RBLOSUM64_{5.0}$	8	1	0.3713
Astral40	1.59	$RBLOSUM64_{5.0}$	8	1	0.3691
Astral40	1.61	$RBLOSUM64_{5.0}$	8	1	0.3714
Astral40	1.63	$RBLOSUM64_{5.0}$	8	1	0.3796
Astral40	1.65	$RBLOSUM64_{5.0}$	8	1	0.3760
Astral40	1.67	$RBLOSUM64_{5.0}$	9	1	0.3815
Astral40	1.69	$RBLOSUM64_{5.0}$	9	1	0.3690
Astral40	1.71	$RBLOSUM64_{5.0}$	9	1	0.3638
Astral40	1.73	$RBLOSUM64_{5.0}$	9	1	0.3674
Astral40	1.75	$RBLOSUM64_{5.0}$	9	1	0.3719
Astral40	2.01	$RBLOSUM64_{5.0}$	9	1	0.3952
Astral40	2.02	$RBLOSUM64_{5.0}$	9	1	0.3952
Astral40	2.03	$RBLOSUM64_{5.0}$	9	1	0.4102
Astral40	2.04	$RBLOSUM64_{5.0}$	9	1	0.4229
Astral40	2.05	$RBLOSUM64_{5.0}$	9	1	0.4250
Astral40	2.06	$RBLOSUM64_{5.0}$	9	1	0.4351

Table B.4: Highest achieved coverage values (Cov.) on the different ASTRAL40 subset versions for the six BLOCKS 5.0-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral40	1.55	BLOSUM50$_{13+}$	14	1	0.3750
Astral40	1.57	BLOSUM50$_{13+}$	16	1	0.3671
Astral40	1.59	BLOSUM50$_{13+}$	14	1	0.3679
Astral40	1.61	BLOSUM50$_{13+}$	14	1	0.3696
Astral40	1.63	BLOSUM50$_{13+}$	15	1	0.3739
Astral40	1.65	BLOSUM50$_{13+}$	15	1	0.3694
Astral40	1.67	BLOSUM50$_{13+}$	18	1	0.3786
Astral40	1.69	BLOSUM50$_{13+}$	14	1	0.3676
Astral40	1.71	BLOSUM50$_{13+}$	17	1	0.3641
Astral40	1.73	BLOSUM50$_{13+}$	17	1	0.3661
Astral40	1.75	BLOSUM50$_{13+}$	12	2	0.3683
Astral40	2.01	BLOSUM50$_{13+}$	12	2	0.3929
Astral40	2.02	BLOSUM50$_{13+}$	12	2	0.3929
Astral40	2.03	BLOSUM50$_{13+}$	14	2	0.4084
Astral40	2.04	BLOSUM50$_{13+}$	16	1	0.4218
Astral40	2.05	BLOSUM50$_{13+}$	18	1	0.4248
Astral40	2.06	BLOSUM50$_{13+}$	16	1	0.4319
Astral40	1.55	BLOSUM62$_{13+}$	13	1	0.3842
Astral40	1.57	BLOSUM62$_{13+}$	11	1	0.3749
Astral40	1.59	BLOSUM62$_{13+}$	12	1	0.3752
Astral40	1.61	BLOSUM62$_{13+}$	12	1	0.3765
Astral40	1.63	BLOSUM62$_{13+}$	13	1	0.3786
Astral40	1.65	BLOSUM62$_{13+}$	13	1	0.3733
Astral40	1.67	BLOSUM62$_{13+}$	13	1	0.3808
Astral40	1.69	BLOSUM62$_{13+}$	9	2	0.3677
Astral40	1.71	BLOSUM62$_{13+}$	15	1	0.3675
Astral40	1.73	BLOSUM62$_{13+}$	10	2	0.3696
Astral40	1.75	BLOSUM62$_{13+}$	10	2	0.3706
Astral40	2.01	BLOSUM62$_{13+}$	13	1	0.3962
Astral40	2.02	BLOSUM62$_{13+}$	13	1	0.3962
Astral40	2.03	BLOSUM62$_{13+}$	10	2	0.4128
Astral40	2.04	BLOSUM62$_{13+}$	10	2	0.4260
Astral40	2.05	BLOSUM62$_{13+}$	13	1	0.4289
Astral40	2.06	BLOSUM62$_{13+}$	10	2	0.4357
Astral40	1.55	CorBLOSUM57$_{13+}$	17	1	0.3777
Astral40	1.57	CorBLOSUM57$_{13+}$	18	1	0.3722
Astral40	1.59	CorBLOSUM57$_{13+}$	17	1	0.3695
Astral40	1.61	CorBLOSUM57$_{13+}$	17	1	0.3713
Astral40	1.63	CorBLOSUM57$_{13+}$	15	1	0.3758
Astral40	1.65	CorBLOSUM57$_{13+}$	17	1	0.3736
Astral40	1.67	CorBLOSUM57$_{13+}$	17	1	0.3836
Astral40	1.69	CorBLOSUM57$_{13+}$	16	1	0.3709
Astral40	1.71	CorBLOSUM57$_{13+}$	19	1	0.3685
Astral40	1.73	CorBLOSUM57$_{13+}$	18	1	0.3718
Astral40	1.75	CorBLOSUM57$_{13+}$	18	1	0.3718
Astral40	2.01	CorBLOSUM57$_{13+}$	18	1	0.3961
Astral40	2.02	CorBLOSUM57$_{13+}$	18	1	0.3961
Astral40	2.03	CorBLOSUM57$_{13+}$	19	1	0.4135
Astral40	2.04	CorBLOSUM57$_{13+}$	19	1	0.4255
Astral40	2.05	CorBLOSUM57$_{13+}$	17	1	0.4289
Astral40	2.06	CorBLOSUM57$_{13+}$	19	1	0.4327

Subset	Version	Matrix	Gop	Ext	Cov.
Astral40	1.55	CorBLOSUM66$_{13+}$	12	1	0.3827
Astral40	1.57	CorBLOSUM66$_{13+}$	11	1	0.3754
Astral40	1.59	CorBLOSUM66$_{13+}$	11	1	0.3740
Astral40	1.61	CorBLOSUM66$_{13+}$	10	1	0.3735
Astral40	1.63	CorBLOSUM66$_{13+}$	14	1	0.3793
Astral40	1.65	CorBLOSUM66$_{13+}$	13	1	0.3738
Astral40	1.67	CorBLOSUM66$_{13+}$	11	2	0.3819
Astral40	1.69	CorBLOSUM66$_{13+}$	13	1	0.3681
Astral40	1.71	CorBLOSUM66$_{13+}$	14	1	0.3656
Astral40	1.73	CorBLOSUM66$_{13+}$	13	1	0.3710
Astral40	1.75	CorBLOSUM66$_{13+}$	9	2	0.3747
Astral40	2.01	CorBLOSUM66$_{13+}$	14	1	0.3980
Astral40	2.02	CorBLOSUM66$_{13+}$	14	1	0.3981
Astral40	2.03	CorBLOSUM66$_{13+}$	12	1	0.4140
Astral40	2.04	CorBLOSUM66$_{13+}$	9	2	0.4273
Astral40	2.05	CorBLOSUM66$_{13+}$	9	2	0.4289
Astral40	2.06	CorBLOSUM66$_{13+}$	13	1	0.4362
Astral40	1.55	RBLOSUM59$_{13+}$	15	1	0.3818
Astral40	1.57	RBLOSUM59$_{13+}$	15	1	0.3729
Astral40	1.59	RBLOSUM59$_{13+}$	16	1	0.3706
Astral40	1.61	RBLOSUM59$_{13+}$	14	1	0.3733
Astral40	1.63	RBLOSUM59$_{13+}$	16	1	0.3798
Astral40	1.65	RBLOSUM59$_{13+}$	16	1	0.3779
Astral40	1.67	RBLOSUM59$_{13+}$	19	1	0.3820
Astral40	1.69	RBLOSUM59$_{13+}$	19	1	0.3710
Astral40	1.71	RBLOSUM59$_{13+}$	19	1	0.3680
Astral40	1.73	RBLOSUM59$_{13+}$	19	1	0.3700
Astral40	1.75	RBLOSUM59$_{13+}$	11	2	0.3719
Astral40	2.01	RBLOSUM59$_{13+}$	12	2	0.3966
Astral40	2.02	RBLOSUM59$_{13+}$	12	2	0.3966
Astral40	2.03	RBLOSUM59$_{13+}$	19	1	0.4112
Astral40	2.04	RBLOSUM59$_{13+}$	18	1	0.4257
Astral40	2.05	RBLOSUM59$_{13+}$	15	1	0.4276
Astral40	2.06	RBLOSUM59$_{13+}$	17	1	0.4335
Astral40	1.55	RBLOSUM69$_{13+}$	11	2	0.3816
Astral40	1.57	RBLOSUM69$_{13+}$	13	1	0.3748
Astral40	1.59	RBLOSUM69$_{13+}$	13	1	0.3766
Astral40	1.61	RBLOSUM69$_{13+}$	11	1	0.3770
Astral40	1.63	RBLOSUM69$_{13+}$	12	1	0.3776
Astral40	1.65	RBLOSUM69$_{13+}$	14	1	0.3757
Astral40	1.67	RBLOSUM69$_{13+}$	14	1	0.3836
Astral40	1.69	RBLOSUM69$_{13+}$	15	1	0.3706
Astral40	1.71	RBLOSUM69$_{13+}$	15	1	0.3672
Astral40	1.73	RBLOSUM69$_{13+}$	14	1	0.3705
Astral40	1.75	RBLOSUM69$_{13+}$	10	2	0.3718
Astral40	2.01	RBLOSUM69$_{13+}$	14	1	0.3966
Astral40	2.02	RBLOSUM69$_{13+}$	14	1	0.3966
Astral40	2.03	RBLOSUM69$_{13+}$	13	1	0.4124
Astral40	2.04	RBLOSUM69$_{13+}$	13	1	0.4265
Astral40	2.05	RBLOSUM69$_{13+}$	13	1	0.4296
Astral40	2.06	RBLOSUM69$_{13+}$	13	1	0.4354

Table B.5: Highest achieved coverage values (Cov.) on the different ASTRAL40 subset versions for the six BLOCKS 13+-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral40	1.55	BLOSUM50$_{14.3}$	16	1	0.3744
Astral40	1.57	BLOSUM50$_{14.3}$	19	1	0.3684
Astral40	1.59	BLOSUM50$_{14.3}$	18	1	0.3679
Astral40	1.61	BLOSUM50$_{14.3}$	17	1	0.3705
Astral40	1.63	BLOSUM50$_{14.3}$	18	1	0.3723
Astral40	1.65	BLOSUM50$_{14.3}$	18	1	0.3705
Astral40	1.67	BLOSUM50$_{14.3}$	19	1	0.3775
Astral40	1.69	BLOSUM50$_{14.3}$	16	2	0.3653
Astral40	1.71	BLOSUM50$_{14.3}$	17	2	0.3627
Astral40	1.73	BLOSUM50$_{14.3}$	16	2	0.3666
Astral40	1.75	BLOSUM50$_{14.3}$	17	2	0.3722
Astral40	2.01	BLOSUM50$_{14.3}$	16	2	0.3925
Astral40	2.02	BLOSUM50$_{14.3}$	16	2	0.3925
Astral40	2.03	BLOSUM50$_{14.3}$	16	2	0.4070
Astral40	2.04	BLOSUM50$_{14.3}$	20	1	0.4237
Astral40	2.05	BLOSUM50$_{14.3}$	20	1	0.4276
Astral40	2.06	BLOSUM50$_{14.3}$	16	2	0.4291
Astral40	1.55	BLOSUM62$_{14.3}$	15	1	0.3765
Astral40	1.57	BLOSUM62$_{14.3}$	16	1	0.3690
Astral40	1.59	BLOSUM62$_{14.3}$	19	1	0.3678
Astral40	1.61	BLOSUM62$_{14.3}$	17	1	0.3719
Astral40	1.63	BLOSUM62$_{14.3}$	18	2	0.3726
Astral40	1.65	BLOSUM62$_{14.3}$	18	2	0.3727
Astral40	1.67	BLOSUM62$_{14.3}$	17	1	0.3790
Astral40	1.69	BLOSUM62$_{14.3}$	17	2	0.3652
Astral40	1.71	BLOSUM62$_{14.3}$	20	1	0.3658
Astral40	1.73	BLOSUM62$_{14.3}$	20	1	0.3689
Astral40	1.75	BLOSUM62$_{14.3}$	20	1	0.3740
Astral40	2.01	BLOSUM62$_{14.3}$	20	1	0.3963
Astral40	2.02	BLOSUM62$_{14.3}$	20	1	0.3963
Astral40	2.03	BLOSUM62$_{14.3}$	19	1	0.4107
Astral40	2.04	BLOSUM62$_{14.3}$	19	1	0.4264
Astral40	2.05	BLOSUM62$_{14.3}$	18	1	0.4296
Astral40	2.06	BLOSUM62$_{14.3}$	18	1	0.4330
Astral40	1.55	CorBLOSUM57$_{14.3}$	18	1	0.3720
Astral40	1.57	CorBLOSUM57$_{14.3}$	19	1	0.3664
Astral40	1.59	CorBLOSUM57$_{14.3}$	17	1	0.3656
Astral40	1.61	CorBLOSUM57$_{14.3}$	17	1	0.3691
Astral40	1.63	CorBLOSUM57$_{14.3}$	19	1	0.3763
Astral40	1.65	CorBLOSUM57$_{14.3}$	19	1	0.3749
Astral40	1.67	CorBLOSUM57$_{14.3}$	16	1	0.3815
Astral40	1.69	CorBLOSUM57$_{14.3}$	19	1	0.3686
Astral40	1.71	CorBLOSUM57$_{14.3}$	20	1	0.3658
Astral40	1.73	CorBLOSUM57$_{14.3}$	20	1	0.3680
Astral40	1.75	CorBLOSUM57$_{14.3}$	20	1	0.3742
Astral40	2.01	CorBLOSUM57$_{14.3}$	19	1	0.3960
Astral40	2.02	CorBLOSUM57$_{14.3}$	19	1	0.3960
Astral40	2.03	CorBLOSUM57$_{14.3}$	15	2	0.4119
Astral40	2.04	CorBLOSUM57$_{14.3}$	18	1	0.4271
Astral40	2.05	CorBLOSUM57$_{14.3}$	18	1	0.4299
Astral40	2.06	CorBLOSUM57$_{14.3}$	19	1	0.4324

Subset	Version	Matrix	Gop	Ext	Cov.
Astral40	1.55	CorBLOSUM67$_{14.3}$	15	1	0.3803
Astral40	1.57	CorBLOSUM67$_{14.3}$	15	1	0.3711
Astral40	1.59	CorBLOSUM67$_{14.3}$	16	1	0.3679
Astral40	1.61	CorBLOSUM67$_{14.3}$	16	1	0.3716
Astral40	1.63	CorBLOSUM67$_{14.3}$	17	1	0.3763
Astral40	1.65	CorBLOSUM67$_{14.3}$	16	1	0.3762
Astral40	1.67	CorBLOSUM67$_{14.3}$	16	2	0.3829
Astral40	1.69	CorBLOSUM67$_{14.3}$	13	2	0.3713
Astral40	1.71	CorBLOSUM67$_{14.3}$	14	2	0.3678
Astral40	1.73	CorBLOSUM67$_{14.3}$	18	1	0.3702
Astral40	1.75	CorBLOSUM67$_{14.3}$	19	1	0.3739
Astral40	2.01	CorBLOSUM67$_{14.3}$	19	1	0.3974
Astral40	2.02	CorBLOSUM67$_{14.3}$	13	2	0.3972
Astral40	2.03	CorBLOSUM67$_{14.3}$	13	2	0.4123
Astral40	2.04	CorBLOSUM67$_{14.3}$	18	1	0.4277
Astral40	2.05	CorBLOSUM67$_{14.3}$	18	1	0.4310
Astral40	2.06	CorBLOSUM67$_{14.3}$	19	1	0.4364
Astral40	1.55	RBLOSUM59$_{14.3}$	16	1	0.3754
Astral40	1.57	RBLOSUM59$_{14.3}$	12	2	0.3694
Astral40	1.59	RBLOSUM59$_{14.3}$	17	1	0.3696
Astral40	1.61	RBLOSUM59$_{14.3}$	17	1	0.3725
Astral40	1.63	RBLOSUM59$_{14.3}$	19	1	0.3773
Astral40	1.65	RBLOSUM59$_{14.3}$	20	1	0.3775
Astral40	1.67	RBLOSUM59$_{14.3}$	19	1	0.3819
Astral40	1.69	RBLOSUM59$_{14.3}$	16	1	0.3690
Astral40	1.71	RBLOSUM59$_{14.3}$	15	2	0.3636
Astral40	1.73	RBLOSUM59$_{14.3}$	19	1	0.3682
Astral40	1.75	RBLOSUM59$_{14.3}$	16	2	0.3734
Astral40	2.01	RBLOSUM59$_{14.3}$	17	2	0.3959
Astral40	2.02	RBLOSUM59$_{14.3}$	17	2	0.3959
Astral40	2.03	RBLOSUM59$_{14.3}$	16	2	0.4095
Astral40	2.04	RBLOSUM59$_{14.3}$	18	1	0.4267
Astral40	2.05	RBLOSUM59$_{14.3}$	17	1	0.4296
Astral40	2.06	RBLOSUM59$_{14.3}$	18	1	0.4328
Astral40	1.55	RBLOSUM69$_{14.3}$	15	1	0.3787
Astral40	1.57	RBLOSUM69$_{14.3}$	17	1	0.3703
Astral40	1.59	RBLOSUM69$_{14.3}$	17	1	0.3717
Astral40	1.61	RBLOSUM69$_{14.3}$	15	1	0.3747
Astral40	1.63	RBLOSUM69$_{14.3}$	17	1	0.3756
Astral40	1.65	RBLOSUM69$_{14.3}$	17	1	0.3758
Astral40	1.67	RBLOSUM69$_{14.3}$	20	1	0.3810
Astral40	1.69	RBLOSUM69$_{14.3}$	18	1	0.3706
Astral40	1.71	RBLOSUM69$_{14.3}$	19	1	0.3668
Astral40	1.73	RBLOSUM69$_{14.3}$	20	1	0.3704
Astral40	1.75	RBLOSUM69$_{14.3}$	15	2	0.3739
Astral40	2.01	RBLOSUM69$_{14.3}$	15	2	0.3971
Astral40	2.02	RBLOSUM69$_{14.3}$	15	2	0.3971
Astral40	2.03	RBLOSUM69$_{14.3}$	19	1	0.4114
Astral40	2.04	RBLOSUM69$_{14.3}$	18	1	0.4272
Astral40	2.05	RBLOSUM69$_{14.3}$	19	1	0.4311
Astral40	2.06	RBLOSUM69$_{14.3}$	16	1	0.4344

Table B.6: Highest achieved coverage values (Cov.) on the different ASTRAL40 subset versions for the six BLOCKS 14.3-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral70	1.55	$BLOSUM50_{5.0}$	14	1	0.4749
Astral70	1.57	$BLOSUM50_{5.0}$	14	1	0.4680
Astral70	1.59	$BLOSUM50_{5.0}$	12	1	0.4647
Astral70	1.61	$BLOSUM50_{5.0}$	12	1	0.4646
Astral70	1.63	$BLOSUM50_{5.0}$	12	1	0.4687
Astral70	1.65	$BLOSUM50_{5.0}$	12	1	0.4725
Astral70	1.67	$BLOSUM50_{5.0}$	14	1	0.4826
Astral70	1.69	$BLOSUM50_{5.0}$	15	1	0.4713
Astral70	1.71	$BLOSUM50_{5.0}$	12	2	0.4582
Astral70	1.73	$BLOSUM50_{5.0}$	8	2	0.4580
Astral70	1.75	$BLOSUM50_{5.0}$	8	2	0.4605
Astral70	2.01	$BLOSUM50_{5.0}$	11	2	0.4933
Astral70	2.02	$BLOSUM50_{5.0}$	11	2	0.4934
Astral70	2.03	$BLOSUM50_{5.0}$	11	2	0.5146
Astral70	2.04	$BLOSUM50_{5.0}$	9	2	0.5285
Astral70	2.05	$BLOSUM50_{5.0}$	11	1	0.5310
Astral70	2.06	$BLOSUM50_{5.0}$	15	1	0.5420
Astral70	1.55	$BLOSUM62_{5.0}$	7	2	0.4748
Astral70	1.57	$BLOSUM62_{5.0}$	9	1	0.4681
Astral70	1.59	$BLOSUM62_{5.0}$	9	1	0.4651
Astral70	1.61	$BLOSUM62_{5.0}$	9	1	0.4649
Astral70	1.63	$BLOSUM62_{5.0}$	9	1	0.4694
Astral70	1.65	$BLOSUM62_{5.0}$	9	1	0.4738
Astral70	1.67	$BLOSUM62_{5.0}$	9	1	0.4820
Astral70	1.69	$BLOSUM62_{5.0}$	9	1	0.4708
Astral70	1.71	$BLOSUM62_{5.0}$	9	1	0.4580
Astral70	1.73	$BLOSUM62_{5.0}$	8	1	0.4540
Astral70	1.75	$BLOSUM62_{5.0}$	9	1	0.4593
Astral70	2.01	$BLOSUM62_{5.0}$	10	1	0.4911
Astral70	2.02	$BLOSUM62_{5.0}$	10	1	0.4911
Astral70	2.03	$BLOSUM62_{5.0}$	9	1	0.5122
Astral70	2.04	$BLOSUM62_{5.0}$	8	1	0.5247
Astral70	2.05	$BLOSUM62_{5.0}$	8	1	0.5271
Astral70	2.06	$BLOSUM62_{5.0}$	9	1	0.5402
Astral70	1.55	$CorBLOSUM49_{5.0}$	13	1	0.4767
Astral70	1.57	$CorBLOSUM49_{5.0}$	13	1	0.4701
Astral70	1.59	$CorBLOSUM49_{5.0}$	13	1	0.4685
Astral70	1.61	$CorBLOSUM49_{5.0}$	13	1	0.4673
Astral70	1.63	$CorBLOSUM49_{5.0}$	14	1	0.4713
Astral70	1.65	$CorBLOSUM49_{5.0}$	14	1	0.4754
Astral70	1.67	$CorBLOSUM49_{5.0}$	14	1	0.4844
Astral70	1.69	$CorBLOSUM49_{5.0}$	15	1	0.4728
Astral70	1.71	$CorBLOSUM49_{5.0}$	15	1	0.4606
Astral70	1.73	$CorBLOSUM49_{5.0}$	9	2	0.4580
Astral70	1.75	$CorBLOSUM49_{5.0}$	13	1	0.4626
Astral70	2.01	$CorBLOSUM49_{5.0}$	15	1	0.4934
Astral70	2.02	$CorBLOSUM49_{5.0}$	15	1	0.4935
Astral70	2.03	$CorBLOSUM49_{5.0}$	15	1	0.5152
Astral70	2.04	$CorBLOSUM49_{5.0}$	14	1	0.5296
Astral70	2.05	$CorBLOSUM49_{5.0}$	14	1	0.5321
Astral70	2.06	$CorBLOSUM49_{5.0}$	11	2	0.5419

Subset	Version	Matrix	Gop	Ext	Cov.
Astral70	1.55	$CorBLOSUM61_{5.0}$	9	1	0.4742
Astral70	1.57	$CorBLOSUM61_{5.0}$	8	1	0.4661
Astral70	1.59	$CorBLOSUM61_{5.0}$	8	1	0.4646
Astral70	1.61	$CorBLOSUM61_{5.0}$	8	1	0.4635
Astral70	1.63	$CorBLOSUM61_{5.0}$	8	1	0.4692
Astral70	1.65	$CorBLOSUM61_{5.0}$	8	1	0.4732
Astral70	1.67	$CorBLOSUM61_{5.0}$	8	1	0.4816
Astral70	1.69	$CorBLOSUM61_{5.0}$	9	1	0.4677
Astral70	1.71	$CorBLOSUM61_{5.0}$	9	1	0.4569
Astral70	1.73	$CorBLOSUM61_{5.0}$	9	1	0.4549
Astral70	1.75	$CorBLOSUM61_{5.0}$	9	1	0.4596
Astral70	2.01	$CorBLOSUM61_{5.0}$	9	1	0.4909
Astral70	2.02	$CorBLOSUM61_{5.0}$	9	1	0.4909
Astral70	2.03	$CorBLOSUM61_{5.0}$	9	1	0.5123
Astral70	2.04	$CorBLOSUM61_{5.0}$	8	1	0.5251
Astral70	2.05	$CorBLOSUM61_{5.0}$	8	1	0.5277
Astral70	2.06	$CorBLOSUM61_{5.0}$	9	1	0.5403
Astral70	1.55	$RBLOSUM52_{5.0}$	13	1	0.4770
Astral70	1.57	$RBLOSUM52_{5.0}$	13	1	0.4706
Astral70	1.59	$RBLOSUM52_{5.0}$	13	1	0.4678
Astral70	1.61	$RBLOSUM52_{5.0}$	13	1	0.4666
Astral70	1.63	$RBLOSUM52_{5.0}$	14	1	0.4724
Astral70	1.65	$RBLOSUM52_{5.0}$	13	1	0.4763
Astral70	1.67	$RBLOSUM52_{5.0}$	13	1	0.4838
Astral70	1.69	$RBLOSUM52_{5.0}$	15	1	0.4726
Astral70	1.71	$RBLOSUM52_{5.0}$	9	2	0.4612
Astral70	1.73	$RBLOSUM52_{5.0}$	9	2	0.4610
Astral70	1.75	$RBLOSUM52_{5.0}$	9	2	0.4636
Astral70	2.01	$RBLOSUM52_{5.0}$	14	1	0.4939
Astral70	2.02	$RBLOSUM52_{5.0}$	14	1	0.4939
Astral70	2.03	$RBLOSUM52_{5.0}$	15	1	0.5157
Astral70	2.04	$RBLOSUM52_{5.0}$	8	2	0.5296
Astral70	2.05	$RBLOSUM52_{5.0}$	10	2	0.5318
Astral70	2.06	$RBLOSUM52_{5.0}$	12	1	0.5420
Astral70	1.55	$RBLOSUM64_{5.0}$	9	1	0.4746
Astral70	1.57	$RBLOSUM64_{5.0}$	8	1	0.4680
Astral70	1.59	$RBLOSUM64_{5.0}$	8	1	0.4652
Astral70	1.61	$RBLOSUM64_{5.0}$	8	1	0.4644
Astral70	1.63	$RBLOSUM64_{5.0}$	8	1	0.4694
Astral70	1.65	$RBLOSUM64_{5.0}$	8	1	0.4727
Astral70	1.67	$RBLOSUM64_{5.0}$	8	1	0.4813
Astral70	1.69	$RBLOSUM64_{5.0}$	9	1	0.4688
Astral70	1.71	$RBLOSUM64_{5.0}$	10	1	0.4577
Astral70	1.73	$RBLOSUM64_{5.0}$	9	1	0.4557
Astral70	1.75	$RBLOSUM64_{5.0}$	9	1	0.4607
Astral70	2.01	$RBLOSUM64_{5.0}$	9	1	0.4922
Astral70	2.02	$RBLOSUM64_{5.0}$	9	1	0.4926
Astral70	2.03	$RBLOSUM64_{5.0}$	9	1	0.5119
Astral70	2.04	$RBLOSUM64_{5.0}$	8	1	0.5242
Astral70	2.05	$RBLOSUM64_{5.0}$	9	1	0.5265
Astral70	2.06	$RBLOSUM64_{5.0}$	9	1	0.5404

Table B.7: Highest achieved coverage values (Cov.) on the different ASTRAL70 subset versions for the six BLOCKS 5.0-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral70	1.55	BLOSUM50$_{13+}$	14	1	0.4704
Astral70	1.57	BLOSUM50$_{13+}$	14	1	0.4635
Astral70	1.59	BLOSUM50$_{13+}$	14	1	0.4611
Astral70	1.61	BLOSUM50$_{13+}$	14	1	0.4618
Astral70	1.63	BLOSUM50$_{13+}$	16	1	0.4657
Astral70	1.65	BLOSUM50$_{13+}$	15	1	0.4695
Astral70	1.67	BLOSUM50$_{13+}$	18	1	0.4804
Astral70	1.69	BLOSUM50$_{13+}$	15	1	0.4692
Astral70	1.71	BLOSUM50$_{13+}$	17	1	0.4597
Astral70	1.73	BLOSUM50$_{13+}$	16	1	0.4581
Astral70	1.75	BLOSUM50$_{13+}$	16	1	0.4615
Astral70	2.01	BLOSUM50$_{13+}$	16	1	0.4940
Astral70	2.02	BLOSUM50$_{13+}$	16	1	0.4941
Astral70	2.03	BLOSUM50$_{13+}$	16	1	0.5151
Astral70	2.04	BLOSUM50$_{13+}$	15	1	0.5300
Astral70	2.05	BLOSUM50$_{13+}$	17	1	0.5324
Astral70	2.06	BLOSUM50$_{13+}$	17	1	0.5413
Astral70	1.55	BLOSUM62$_{13+}$	12	1	0.4739
Astral70	1.57	BLOSUM62$_{13+}$	11	1	0.4684
Astral70	1.59	BLOSUM62$_{13+}$	12	1	0.4635
Astral70	1.61	BLOSUM62$_{13+}$	12	1	0.4640
Astral70	1.63	BLOSUM62$_{13+}$	13	1	0.4672
Astral70	1.65	BLOSUM62$_{13+}$	13	1	0.4717
Astral70	1.67	BLOSUM62$_{13+}$	14	1	0.4815
Astral70	1.69	BLOSUM62$_{13+}$	10	2	0.4701
Astral70	1.71	BLOSUM62$_{13+}$	13	1	0.4599
Astral70	1.73	BLOSUM62$_{13+}$	9	2	0.4574
Astral70	1.75	BLOSUM62$_{13+}$	10	2	0.4624
Astral70	2.01	BLOSUM62$_{13+}$	10	2	0.4925
Astral70	2.02	BLOSUM62$_{13+}$	10	2	0.4933
Astral70	2.03	BLOSUM62$_{13+}$	10	2	0.5154
Astral70	2.04	BLOSUM62$_{13+}$	10	2	0.5314
Astral70	2.05	BLOSUM62$_{13+}$	12	1	0.5335
Astral70	2.06	BLOSUM62$_{13+}$	13	1	0.5418
Astral70	1.55	CorBLOSUM57$_{13+}$	17	1	0.4730
Astral70	1.57	CorBLOSUM57$_{13+}$	16	1	0.4668
Astral70	1.59	CorBLOSUM57$_{13+}$	17	1	0.4632
Astral70	1.61	CorBLOSUM57$_{13+}$	17	1	0.4640
Astral70	1.63	CorBLOSUM57$_{13+}$	17	1	0.4685
Astral70	1.65	CorBLOSUM57$_{13+}$	17	1	0.4717
Astral70	1.67	CorBLOSUM57$_{13+}$	17	1	0.4830
Astral70	1.69	CorBLOSUM57$_{13+}$	16	1	0.4698
Astral70	1.71	CorBLOSUM57$_{13+}$	18	1	0.4626
Astral70	1.73	CorBLOSUM57$_{13+}$	19	1	0.4604
Astral70	1.75	CorBLOSUM57$_{13+}$	18	1	0.4634
Astral70	2.01	CorBLOSUM57$_{13+}$	18	1	0.4962
Astral70	2.02	CorBLOSUM57$_{13+}$	18	1	0.4962
Astral70	2.03	CorBLOSUM57$_{13+}$	17	1	0.5175
Astral70	2.04	CorBLOSUM57$_{13+}$	18	1	0.5316
Astral70	2.05	CorBLOSUM57$_{13+}$	12	2	0.5340
Astral70	2.06	CorBLOSUM57$_{13+}$	17	1	0.5432

Subset	Version	Matrix	Gop	Ext	Cov.
Astral70	1.55	CorBLOSUM66$_{13+}$	11	1	0.4762
Astral70	1.57	CorBLOSUM66$_{13+}$	11	1	0.4704
Astral70	1.59	CorBLOSUM66$_{13+}$	12	1	0.4665
Astral70	1.61	CorBLOSUM66$_{13+}$	11	1	0.4654
Astral70	1.63	CorBLOSUM66$_{13+}$	14	1	0.4678
Astral70	1.65	CorBLOSUM66$_{13+}$	15	1	0.4703
Astral70	1.67	CorBLOSUM66$_{13+}$	15	1	0.4811
Astral70	1.69	CorBLOSUM66$_{13+}$	13	1	0.4692
Astral70	1.71	CorBLOSUM66$_{13+}$	14	1	0.4612
Astral70	1.73	CorBLOSUM66$_{13+}$	12	1	0.4592
Astral70	1.75	CorBLOSUM66$_{13+}$	9	2	0.4640
Astral70	2.01	CorBLOSUM66$_{13+}$	9	2	0.4962
Astral70	2.02	CorBLOSUM66$_{13+}$	9	2	0.4962
Astral70	2.03	CorBLOSUM66$_{13+}$	12	1	0.5168
Astral70	2.04	CorBLOSUM66$_{13+}$	12	1	0.5326
Astral70	2.05	CorBLOSUM66$_{13+}$	12	1	0.5340
Astral70	2.06	CorBLOSUM66$_{13+}$	12	1	0.5445
Astral70	1.55	RBLOSUM59$_{13+}$	17	1	0.4727
Astral70	1.57	RBLOSUM59$_{13+}$	17	1	0.4677
Astral70	1.59	RBLOSUM59$_{13+}$	15	1	0.4646
Astral70	1.61	RBLOSUM59$_{13+}$	15	1	0.4649
Astral70	1.63	RBLOSUM59$_{13+}$	16	1	0.4698
Astral70	1.65	RBLOSUM59$_{13+}$	16	1	0.4724
Astral70	1.67	RBLOSUM59$_{13+}$	17	1	0.4826
Astral70	1.69	RBLOSUM59$_{13+}$	19	1	0.4714
Astral70	1.71	RBLOSUM59$_{13+}$	17	1	0.4639
Astral70	1.73	RBLOSUM59$_{13+}$	18	1	0.4604
Astral70	1.75	RBLOSUM59$_{13+}$	16	1	0.4635
Astral70	2.01	RBLOSUM59$_{13+}$	16	1	0.4957
Astral70	2.02	RBLOSUM59$_{13+}$	16	1	0.4958
Astral70	2.03	RBLOSUM59$_{13+}$	16	1	0.5166
Astral70	2.04	RBLOSUM59$_{13+}$	18	1	0.5320
Astral70	2.05	RBLOSUM59$_{13+}$	17	1	0.5344
Astral70	2.06	RBLOSUM59$_{13+}$	17	1	0.5429
Astral70	1.55	RBLOSUM69$_{13+}$	13	1	0.4751
Astral70	1.57	RBLOSUM69$_{13+}$	12	1	0.4681
Astral70	1.59	RBLOSUM69$_{13+}$	12	1	0.4663
Astral70	1.61	RBLOSUM69$_{13+}$	12	1	0.4656
Astral70	1.63	RBLOSUM69$_{13+}$	13	1	0.4686
Astral70	1.65	RBLOSUM69$_{13+}$	14	1	0.4706
Astral70	1.67	RBLOSUM69$_{13+}$	15	1	0.4804
Astral70	1.69	RBLOSUM69$_{13+}$	10	2	0.4713
Astral70	1.71	RBLOSUM69$_{13+}$	10	2	0.4616
Astral70	1.73	RBLOSUM69$_{13+}$	12	1	0.4595
Astral70	1.75	RBLOSUM69$_{13+}$	9	2	0.4622
Astral70	2.01	RBLOSUM69$_{13+}$	13	1	0.4934
Astral70	2.02	RBLOSUM69$_{13+}$	9	2	0.4941
Astral70	2.03	RBLOSUM69$_{13+}$	9	2	0.5146
Astral70	2.04	RBLOSUM69$_{13+}$	9	2	0.5313
Astral70	2.05	RBLOSUM69$_{13+}$	12	1	0.5342
Astral70	2.06	RBLOSUM69$_{13+}$	12	1	0.5428

Table B.8: Highest achieved coverage values (Cov.) on the different ASTRAL70 subset versions for the six BLOCKS 13+-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Version	Matrix	Gop	Ext	Cov.
Astral70	1.55	$BLOSUM50_{14.3}$	18	1	0.4710
Astral70	1.57	$BLOSUM50_{14.3}$	18	1	0.4654
Astral70	1.59	$BLOSUM50_{14.3}$	19	1	0.4617
Astral70	1.61	$BLOSUM50_{14.3}$	19	1	0.4610
Astral70	1.63	$BLOSUM50_{14.3}$	18	1	0.4617
Astral70	1.65	$BLOSUM50_{14.3}$	20	1	0.4653
Astral70	1.67	$BLOSUM50_{14.3}$	19	1	0.4773
Astral70	1.69	$BLOSUM50_{14.3}$	16	2	0.4665
Astral70	1.71	$BLOSUM50_{14.3}$	16	2	0.4607
Astral70	1.73	$BLOSUM50_{14.3}$	16	2	0.4579
Astral70	1.75	$BLOSUM50_{14.3}$	16	2	0.4624
Astral70	2.01	$BLOSUM50_{14.3}$	16	2	0.4943
Astral70	2.02	$BLOSUM50_{14.3}$	16	2	0.4943
Astral70	2.03	$BLOSUM50_{14.3}$	16	2	0.5160
Astral70	2.04	$BLOSUM50_{14.3}$	16	2	0.5331
Astral70	2.05	$BLOSUM50_{14.3}$	16	2	0.5349
Astral70	2.06	$BLOSUM50_{14.3}$	16	2	0.5401
Astral70	1.55	$BLOSUM62_{14.3}$	16	1	0.4740
Astral70	1.57	$BLOSUM62_{14.3}$	16	1	0.4672
Astral70	1.59	$BLOSUM62_{14.3}$	17	1	0.4644
Astral70	1.61	$BLOSUM62_{14.3}$	16	1	0.4634
Astral70	1.63	$BLOSUM62_{14.3}$	19	1	0.4638
Astral70	1.65	$BLOSUM62_{14.3}$	16	2	0.4673
Astral70	1.67	$BLOSUM62_{14.3}$	20	1	0.4790
Astral70	1.69	$BLOSUM62_{14.3}$	15	2	0.4683
Astral70	1.71	$BLOSUM62_{14.3}$	20	1	0.4622
Astral70	1.73	$BLOSUM62_{14.3}$	19	1	0.4599
Astral70	1.75	$BLOSUM62_{14.3}$	19	1	0.4650
Astral70	2.01	$BLOSUM62_{14.3}$	19	1	0.4964
Astral70	2.02	$BLOSUM62_{14.3}$	19	1	0.4964
Astral70	2.03	$BLOSUM62_{14.3}$	19	1	0.5174
Astral70	2.04	$BLOSUM62_{14.3}$	14	2	0.5334
Astral70	2.05	$BLOSUM62_{14.3}$	17	1	0.5349
Astral70	2.06	$BLOSUM62_{14.3}$	17	1	0.5422
Astral70	1.55	$CorBLOSUM57_{14.3}$	18	1	0.4714
Astral70	1.57	$CorBLOSUM57_{14.3}$	17	1	0.4666
Astral70	1.59	$CorBLOSUM57_{14.3}$	18	1	0.4608
Astral70	1.61	$CorBLOSUM57_{14.3}$	20	1	0.4591
Astral70	1.63	$CorBLOSUM57_{14.3}$	18	1	0.4643
Astral70	1.65	$CorBLOSUM57_{14.3}$	20	1	0.4694
Astral70	1.67	$CorBLOSUM57_{14.3}$	20	1	0.4803
Astral70	1.69	$CorBLOSUM57_{14.3}$	20	1	0.4699
Astral70	1.71	$CorBLOSUM57_{14.3}$	20	1	0.4645
Astral70	1.73	$CorBLOSUM57_{14.3}$	15	2	0.4594
Astral70	1.75	$CorBLOSUM57_{14.3}$	15	2	0.4649
Astral70	2.01	$CorBLOSUM57_{14.3}$	15	2	0.4969
Astral70	2.02	$CorBLOSUM57_{14.3}$	15	2	0.4969
Astral70	2.03	$CorBLOSUM57_{14.3}$	15	2	0.5183
Astral70	2.04	$CorBLOSUM57_{14.3}$	19	1	0.5345
Astral70	2.05	$CorBLOSUM57_{14.3}$	18	1	0.5359
Astral70	2.06	$CorBLOSUM57_{14.3}$	15	2	0.5409

Subset	Version	Matrix	Gop	Ext	Cov.
Astral70	1.55	$CorBLOSUM67_{14.3}$	15	1	0.4761
Astral70	1.57	$CorBLOSUM67_{14.3}$	15	1	0.4684
Astral70	1.59	$CorBLOSUM67_{14.3}$	15	1	0.4645
Astral70	1.61	$CorBLOSUM67_{14.3}$	16	1	0.4634
Astral70	1.63	$CorBLOSUM67_{14.3}$	18	1	0.4659
Astral70	1.65	$CorBLOSUM67_{14.3}$	19	1	0.4704
Astral70	1.67	$CorBLOSUM67_{14.3}$	20	1	0.4815
Astral70	1.69	$CorBLOSUM67_{14.3}$	15	2	0.4708
Astral70	1.71	$CorBLOSUM67_{14.3}$	14	2	0.4633
Astral70	1.73	$CorBLOSUM67_{14.3}$	16	1	0.4593
Astral70	1.75	$CorBLOSUM67_{14.3}$	17	1	0.4648
Astral70	2.01	$CorBLOSUM67_{14.3}$	17	1	0.4959
Astral70	2.02	$CorBLOSUM67_{14.3}$	17	1	0.4959
Astral70	2.03	$CorBLOSUM67_{14.3}$	17	1	0.5180
Astral70	2.04	$CorBLOSUM67_{14.3}$	18	1	0.5329
Astral70	2.05	$CorBLOSUM67_{14.3}$	18	1	0.5351
Astral70	2.06	$CorBLOSUM67_{14.3}$	17	1	0.5423
Astral70	1.55	$RBLOSUM59_{14.3}$	17	1	0.4731
Astral70	1.57	$RBLOSUM59_{14.3}$	17	1	0.4680
Astral70	1.59	$RBLOSUM59_{14.3}$	17	1	0.4649
Astral70	1.61	$RBLOSUM59_{14.3}$	17	1	0.4650
Astral70	1.63	$RBLOSUM59_{14.3}$	18	1	0.4650
Astral70	1.65	$RBLOSUM59_{14.3}$	18	1	0.4705
Astral70	1.67	$RBLOSUM59_{14.3}$	20	1	0.4808
Astral70	1.69	$RBLOSUM59_{14.3}$	18	1	0.4701
Astral70	1.71	$RBLOSUM59_{14.3}$	18	2	0.4648
Astral70	1.73	$RBLOSUM59_{14.3}$	19	1	0.4598
Astral70	1.75	$RBLOSUM59_{14.3}$	19	1	0.4640
Astral70	2.01	$RBLOSUM59_{14.3}$	18	1	0.4959
Astral70	2.02	$RBLOSUM59_{14.3}$	18	1	0.4959
Astral70	2.03	$RBLOSUM59_{14.3}$	19	1	0.5182
Astral70	2.04	$RBLOSUM59_{14.3}$	15	2	0.5345
Astral70	2.05	$RBLOSUM59_{14.3}$	15	2	0.5371
Astral70	2.06	$RBLOSUM59_{14.3}$	15	2	0.5420
Astral70	1.55	$RBLOSUM69_{14.3}$	15	1	0.4744
Astral70	1.57	$RBLOSUM69_{14.3}$	16	1	0.4680
Astral70	1.59	$RBLOSUM69_{14.3}$	16	1	0.4658
Astral70	1.61	$RBLOSUM69_{14.3}$	15	1	0.4654
Astral70	1.63	$RBLOSUM69_{14.3}$	15	1	0.4660
Astral70	1.65	$RBLOSUM69_{14.3}$	15	1	0.4708
Astral70	1.67	$RBLOSUM69_{14.3}$	20	1	0.4798
Astral70	1.69	$RBLOSUM69_{14.3}$	20	1	0.4694
Astral70	1.71	$RBLOSUM69_{14.3}$	14	2	0.4614
Astral70	1.73	$RBLOSUM69_{14.3}$	17	1	0.4602
Astral70	1.75	$RBLOSUM69_{14.3}$	17	1	0.4648
Astral70	2.01	$RBLOSUM69_{14.3}$	17	1	0.4970
Astral70	2.02	$RBLOSUM69_{14.3}$	17	1	0.4970
Astral70	2.03	$RBLOSUM69_{14.3}$	18	1	0.5181
Astral70	2.04	$RBLOSUM69_{14.3}$	14	2	0.5344
Astral70	2.05	$RBLOSUM69_{14.3}$	14	2	0.5359
Astral70	2.06	$RBLOSUM69_{14.3}$	20	1	0.5424

Table B.9: Highest achieved coverage values (Cov.) on the different ASTRAL70 subset versions for the six BLOCKS 14.3-based substitution matrices tested in the CorBLOSUM benchmark and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Appendix C
Supplemental material - PFASUM substitution matrices

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	J	X	*
A	4	-1	-1	-1	1	0	-1	0	-2	-1	-1	-1	0	-2	-1	1	0	-2	-2	0	-5	-5	-5	-18	-6
R	-1	7	0	0	-3	2	1	-2	1	-4	-3	3	-2	-4	-1	0	0	-2	-2	-3	-4	-2	-7	-18	-6
N	-1	0	7	3	-3	1	1	1	1	-4	-4	1	-3	-4	-1	1	1	-4	-2	-4	-1	-3	-8	-18	-6
D	-1	0	3	8	-4	1	3	0	0	-6	-5	1	-4	-5	0	1	0	-5	-3	-5	-1	-1	-9	-18	-6
C	1	-3	-3	-4	16	-3	-4	-2	-2	0	0	-4	0	0	-3	0	-1	-2	-1	1	-7	-8	-4	-18	-6
Q	0	2	1	1	-3	5	3	-1	1	-3	-3	2	-1	-4	-1	0	0	-3	-2	-3	-3	-2	-7	-18	-6
E	-1	1	1	3	-4	3	6	-1	0	-4	-4	2	-3	-5	0	0	0	-4	-3	-4	-2	-2	-8	-18	-6
G	0	-2	1	0	-2	-1	-1	9	-2	-4	-4	-1	-3	-4	-1	1	-1	-3	-3	-3	-4	-5	-8	-18	-6
H	-2	1	1	0	-2	1	0	-2	12	-4	-3	0	-2	-2	-1	0	-1	-1	2	-3	-3	-4	-7	-18	-6
I	-1	-4	-4	-6	0	-3	-4	-4	-4	5	3	-4	2	2	-3	-3	-1	-1	-1	4	-9	-8	-2	-18	-6
L	-1	-3	-4	-5	0	-3	-4	-4	-3	3	5	-4	3	2	-3	-3	-2	0	0	2	-9	-8	-2	-18	-6
K	-1	3	1	1	-4	2	2	-1	0	-4	-4	6	-2	-4	0	0	0	-4	-2	-3	-3	-2	-8	-18	-6
M	0	-2	-3	-4	0	-1	-3	-3	-2	2	3	-2	6	2	-3	-2	-1	0	0	1	-7	-6	-1	-17	-6
F	-2	-4	-4	-5	0	-4	-5	-4	-2	2	2	-4	2	7	-3	-3	-2	3	4	1	-9	-8	-2	-18	-6
P	-1	-1	-1	0	-3	-1	0	-1	-1	-3	-3	0	-3	-3	10	0	-1	-3	-3	-2	-4	-5	-7	-18	-6
S	1	0	1	1	0	0	0	1	0	-3	-3	0	-2	-3	0	4	2	-3	-2	-2	-3	-4	-7	-18	-6
T	0	0	1	0	-1	0	0	-1	-1	-1	-2	0	-1	-2	-1	2	5	-3	-2	0	-4	-4	-6	-18	-6
W	-2	-2	-4	-5	-2	-3	-4	-3	-1	-1	0	-4	0	3	-3	-3	-3	16	4	-1	-8	-8	-4	-18	-6
Y	-2	-2	-2	-3	-1	-2	-3	-3	2	-1	0	-2	0	4	-3	-2	-2	4	9	-1	-7	-6	-4	-18	-6
V	0	-3	-4	-5	1	-3	-4	-3	-3	4	2	-3	1	1	-2	-2	0	-1	-1	5	-8	-7	-1	-18	-6
B	-5	-4	-1	-1	-7	-3	-2	-4	-3	-9	-9	-3	-7	-9	-4	-3	-4	-8	-7	-8	-1	-6	-13	-22	-6
Z	-5	-2	-3	-1	-8	-2	-2	-5	-4	-8	-8	-2	-6	-8	-5	-4	-4	-8	-6	-7	-6	-2	-12	-22	-6
J	-5	-7	-8	-9	-4	-7	-8	-8	-7	-2	-2	-8	-1	-2	-7	-7	-6	-4	-4	-1	-13	-12	-3	-22	-6
X	-18	-18	-18	-18	-18	-18	-18	-18	-18	-18	-18	-18	-17	-18	-18	-18	-18	-18	-18	-18	-22	-22	-22	-27	-6
*	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	1

Table C.1: The PFASUM31 matrix derived from the Pfam seed alignments of Pfam release 29.0 [Finn et al. 2016]. PFASUM31 has a relative entropy of $H = 0.2297$. The entries are rounded and scaled to 1/4 bit units.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	J	X	*
A	4	-1	-1	-1	0	0	-1	0	-2	-1	-1	-1	0	-2	-1	1	0	-2	-2	0	-4	-4	-4	-13	-5
R	-1	6	0	0	-3	2	1	-2	1	-3	-3	3	-2	-3	-1	0	0	-2	-2	-3	-3	-2	-6	-13	-5
N	-1	0	6	2	-2	1	1	0	1	-4	-4	1	-2	-3	-1	1	0	-3	-2	-3	0	-2	-7	-13	-5
D	-1	0	2	6	-4	1	3	0	0	-5	-5	0	-4	-5	0	0	0	-4	-3	-4	0	-1	-8	-14	-5
C	0	-3	-2	-4	13	-3	-4	-2	-2	-1	-1	-4	0	-1	-3	0	-1	-2	-1	0	-6	-7	-4	-14	-5
Q	0	2	1	1	-3	5	2	-1	1	-3	-3	2	-1	-3	-1	0	0	-3	-2	-2	-2	-1	-6	-13	-5
E	-1	1	1	3	-4	2	5	-1	0	-4	-4	2	-3	-4	-1	0	0	-4	-3	-3	-1	-1	-7	-13	-5
G	0	-2	0	0	-2	-1	-1	7	-2	-4	-4	-1	-3	-4	-1	0	-1	-3	-3	-3	-3	-4	-7	-14	-5
H	-2	1	1	0	-2	1	0	-2	9	-3	-3	0	-2	-1	-1	0	-1	-1	2	-3	-3	-3	-6	-13	-5
I	-1	-3	-4	-5	-1	-3	-4	-4	-3	5	2	-3	2	1	-3	-3	-1	-1	-1	3	-7	-7	-1	-13	-5
L	-1	-3	-4	-5	-1	-3	-4	-4	-3	2	4	-3	2	2	-3	-3	-2	0	0	2	-7	-6	-1	-14	-5
K	-1	3	1	0	-4	2	2	-1	0	-3	-3	5	-2	-4	-1	0	0	-3	-2	-3	-2	-1	-6	-13	-5
M	0	-2	-2	-4	0	-1	-3	-3	-2	2	2	-2	6	1	-3	-2	-1	0	0	1	-6	-5	-1	-13	-5
F	-2	-3	-3	-5	-1	-3	-4	-4	-1	1	2	-4	1	7	-3	-3	-2	3	4	0	-7	-7	-1	-13	-5
P	-1	-1	-1	0	-3	-1	-1	-1	-1	-3	-3	-1	-3	-3	9	0	-1	-3	-3	-2	-4	-4	-6	-14	-5
S	1	0	1	0	0	0	0	0	0	-3	-3	0	-2	-3	0	4	2	-3	-2	-2	-2	-3	-6	-13	-5
T	0	0	0	0	-1	0	0	-1	-1	-1	-2	0	-1	-2	-1	2	4	-3	-2	0	-3	-3	-4	-13	-5
W	-2	-2	-3	-4	-2	-3	-4	-3	-1	-1	0	-3	0	3	-3	-3	-3	13	3	-2	-7	-6	-4	-14	-5
Y	-2	-2	-2	-3	-1	-2	-3	-3	2	-1	0	-2	0	4	-3	-2	-2	3	8	-1	-5	-5	-4	-14	-5
V	0	-3	-3	-4	0	-2	-3	-3	-3	3	2	-3	1	0	-2	-2	0	-2	-1	4	-7	-6	-1	-13	-5
B	-4	-3	0	0	-6	-2	-1	-3	-3	-7	-7	-2	-6	-7	-4	-2	-3	-7	-5	-7	0	-5	-10	-16	-5
Z	-4	-2	-2	-1	-7	-1	-1	-4	-3	-7	-6	-1	-5	-7	-4	-3	-3	-6	-5	-6	-5	-1	-9	-16	-5
J	-4	-6	-7	-8	-4	-6	-7	-7	-6	-1	-1	-6	-1	-1	-6	-6	-4	-4	-4	-1	-10	-9	-1	-17	-5
X	-13	-13	-13	-14	-14	-13	-13	-14	-13	-13	-14	-13	-13	-13	-14	-13	-13	-14	-14	-13	-16	-16	-17	-20	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Table C.2: The PFASUM43 matrix derived from the Pfam seed alignments of Pfam release 29.0 [Finn et al. 2016]. PFASUM43 has a relative entropy of $H = 0.3354$. The entries are rounded and scaled to 1/3 bit units.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	J	X	*
A	5	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-3	0	-5	-4	-4	-14	-6
R	-1	7	0	-1	-4	2	0	-2	1	-4	-3	3	-2	-4	-2	-1	-1	-3	-2	-3	-4	-2	-7	-14	-6
N	-2	0	7	2	-3	1	0	0	1	-5	-4	1	-3	-4	-1	1	0	-4	-2	-4	0	-2	-8	-13	-6
D	-2	-1	2	7	-5	1	3	-1	0	-6	-6	0	-4	-6	-1	0	-1	-5	-4	-5	1	-1	-9	-14	-6
C	0	-4	-3	-5	14	-4	-5	-2	-2	-1	-1	-4	-1	-1	-4	0	-1	-2	-1	0	-7	-7	-4	-14	-6
Q	-1	2	1	1	-4	6	2	-2	1	-4	-3	2	-1	-4	-1	0	0	-3	-2	-3	-2	0	-6	-13	-6
E	-1	0	0	3	-5	2	6	-2	0	-5	-4	1	-3	-5	-1	0	-1	-5	-3	-4	-1	0	-8	-14	-6
G	0	-2	0	-1	-2	-2	-2	8	-2	-5	-5	-2	-4	-5	-2	0	-2	-4	-4	-4	-3	-5	-8	-14	-6
H	-2	1	1	0	-2	1	0	-2	10	-4	-3	0	-2	-1	-2	-1	-1	-1	2	-3	-3	-3	-6	-13	-6
I	-1	-4	-5	-6	-1	-4	-5	-5	-4	6	3	-4	2	1	-4	-3	-1	-2	-2	4	-8	-7	0	-14	-6
L	-1	-3	-4	-6	-1	-3	-4	-5	-3	3	5	-4	3	2	-4	-4	-2	-1	-1	1	-8	-7	-1	-14	-6
K	-1	3	1	0	-4	2	1	-2	0	-4	-4	6	-2	-5	-1	0	0	-4	-3	-3	-3	-1	-7	-13	-6
M	-1	-2	-3	-4	-1	-1	-3	-4	-2	2	3	-2	8	1	-4	-2	-1	-1	-1	1	-7	-5	-1	-13	-6
F	-2	-4	-4	-6	-1	-4	-5	-5	-1	1	2	-5	1	8	-4	-3	-3	3	4	0	-8	-8	-2	-14	-6
P	-1	-2	-1	-1	-4	-1	-1	-2	-2	-4	-4	-1	-4	-4	10	0	-1	-4	-4	-3	-4	-4	-7	-14	-6
S	1	-1	1	0	0	0	0	0	-1	-3	-4	0	-2	-3	0	5	2	-4	-3	-2	-2	-3	-7	-13	-6
T	0	-1	0	-1	-1	0	-1	-2	-1	-1	-2	0	-1	-3	-1	2	6	-3	-2	0	-3	-3	-5	-13	-6
W	-3	-3	-4	-5	-2	-3	-5	-4	-1	-2	-1	-4	-1	3	-4	-4	-3	14	3	-2	-8	-7	-4	-14	-6
Y	-3	-2	-2	-4	-1	-2	-3	-4	2	-2	-1	-3	-1	4	-4	-3	-2	3	9	-2	-6	-6	-4	-14	-6
V	0	-3	-4	-5	0	-3	-4	-4	-3	4	1	-3	1	0	-3	-2	0	-2	-2	5	-8	-6	-1	-14	-6
B	-5	-4	0	1	-7	-2	-1	-3	-3	-8	-8	-3	-7	-8	-4	-2	-3	-8	-6	-8	1	-5	-11	-17	-6
Z	-4	-2	-2	-1	-7	0	0	-5	-3	-7	-7	-1	-5	-8	-4	-3	-3	-7	-6	-6	-5	0	-10	-16	-6
J	-4	-7	-8	-9	-4	-6	-8	-8	-6	0	-1	-7	-1	-2	-7	-7	-5	-4	-4	-1	-11	-10	0	-17	-6
X	-14	-14	-13	-14	-14	-13	-14	-14	-13	-14	-14	-13	-13	-14	-14	-13	-13	-14	-14	-14	-17	-16	-17	-21	-6
*	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	-6	1

Table C.3: The PFASUM60 matrix derived from the Pfam seed alignments of Pfam release 29.0 [Finn et al. 2016]. PFASUM60 has a relative entropy of $H = 0.4941$. The entries are rounded and scaled to 1/3 bit units.

Matrix	Scale	Rel. entropy
PFASUM11	1/6	0.1036
PFASUM12	1/8	0.0668
PFASUM13	1/7	0.0727
PFASUM14	1/7	0.0803
PFASUM15	1/6	0.0953
PFASUM16	1/6	0.1067
PFASUM17	1/6	0.1183
PFASUM18	1/6	0.1239
PFASUM19	1/6	0.1301
PFASUM20	1/5	0.1350
PFASUM21	1/5	0.1418
PFASUM22	1/5	0.1506
PFASUM23	1/5	0.1587
PFASUM24	1/5	0.1675
PFASUM25	1/5	0.1759
PFASUM26	1/5	0.1849
PFASUM27	1/5	0.1938
PFASUM28	1/4	0.2029
PFASUM29	1/4	0.2121
PFASUM30	1/4	0.2211
PFASUM31	1/4	0.2297
PFASUM32	1/4	0.2381
PFASUM33	1/4	0.2466
PFASUM34	1/4	0.2551
PFASUM35	1/4	0.2634
PFASUM36	1/4	0.2716
PFASUM37	1/4	0.2806
PFASUM38	1/4	0.2900
PFASUM39	1/4	0.2995
PFASUM40	1/4	0.3083
PFASUM41	1/4	0.3177
PFASUM42	1/4	0.3265
PFASUM43	1/3	0.3354
PFASUM44	1/3	0.3441
PFASUM45	1/3	0.3529
PFASUM46	1/3	0.3619
PFASUM47	1/3	0.3712
PFASUM48	1/3	0.3803
PFASUM49	1/3	0.3900
PFASUM50	1/3	0.3981
PFASUM51	1/3	0.4084
PFASUM52	1/3	0.4173
PFASUM53	1/3	0.4266
PFASUM54	1/3	0.4362
PFASUM55	1/3	0.4454
PFASUM56	1/3	0.4552
PFASUM57	1/3	0.4654
PFASUM58	1/3	0.4750
PFASUM59	1/3	0.4849
PFASUM60	1/3	0.4941
PFASUM61	1/3	0.5043
PFASUM62	1/3	0.5141
PFASUM63	1/3	0.5241
PFASUM64	1/3	0.5341
PFASUM65	1/3	0.5443
PFASUM66	1/3	0.5547
PFASUM67	1/3	0.5649
PFASUM68	1/3	0.5753
PFASUM69	1/3	0.5860
PFASUM70	1/3	0.5965
PFASUM71	1/3	0.6078
PFASUM72	1/3	0.6187
PFASUM73	1/3	0.6299
PFASUM74	1/2	0.6415
PFASUM75	1/2	0.6528
PFASUM76	1/2	0.6661
PFASUM77	1/2	0.6794
PFASUM78	1/2	0.6931
PFASUM79	1/2	0.7083
PFASUM80	1/2	0.7236
PFASUM81	1/2	0.7288
PFASUM82	1/2	0.7300
PFASUM83	1/2	0.7306
PFASUM84	1/2	0.7308
PFASUM85	1/2	0.7309
PFASUM86	1/2	0.7311
PFASUM87	1/2	0.7311
PFASUM88	1/2	0.7312
PFASUM89	1/2	0.7313
PFASUM90	1/2	0.7314
PFASUM91	1/2	0.7315
PFASUM92	1/2	0.7315
PFASUM93	1/2	0.7315
PFASUM94	1/2	0.7316
PFASUM95	1/2	0.7317
PFASUM96	1/2	0.7317
PFASUM97	1/2	0.7318
PFASUM98	1/2	0.7318
PFASUM99	1/2	0.7319
PFASUM100	1/2	0.7122

Table C.4: List of PFASUM matrices and their corresponding scale and relative entropies.

Subset	Matrix	Gop	Ext	Cov.
ASTRAL20	PFASUM11	18	3	0.0248
ASTRAL20	PFASUM12	20	3	0.0467
ASTRAL20	PFASUM13	20	3	0.0939
ASTRAL20	PFASUM14	20	3	0.1070
ASTRAL20	PFASUM15	20	3	0.1214
ASTRAL20	PFASUM16	17	3	0.1218
ASTRAL20	PFASUM17	20	3	0.1295
ASTRAL20	PFASUM18	20	3	0.1284
ASTRAL20	PFASUM19	17	3	0.1382
ASTRAL20	PFASUM20	14	3	0.1404
ASTRAL20	PFASUM21	18	2	0.1409
ASTRAL20	PFASUM22	15	3	0.1427
ASTRAL20	PFASUM23	14	3	0.1438
ASTRAL20	PFASUM24	18	3	0.1419
ASTRAL20	PFASUM25	20	2	0.1511
ASTRAL20	PFASUM26	19	2	0.1527
ASTRAL20	PFASUM27	20	2	0.1598
ASTRAL20	PFASUM28	15	2	0.1571
ASTRAL20	PFASUM29	14	2	0.1575
ASTRAL20	PFASUM30	20	1	0.1569
ASTRAL20	PFASUM31	15	2	0.1562
ASTRAL20	PFASUM32	20	1	0.1564
ASTRAL20	PFASUM33	20	1	0.1578
ASTRAL20	PFASUM34	20	1	0.1659
ASTRAL20	PFASUM35	15	2	0.1644
ASTRAL20	PFASUM36	15	2	0.1634
ASTRAL20	PFASUM37	15	2	0.1633
ASTRAL20	PFASUM38	20	1	0.1651
ASTRAL20	PFASUM39	16	2	0.1656
ASTRAL20	PFASUM40	16	2	0.1680
ASTRAL20	PFASUM41	17	2	0.1666
ASTRAL20	PFASUM42	16	2	0.1673
ASTRAL20	PFASUM43	14	1	0.1650
ASTRAL20	PFASUM44	14	1	0.1671
ASTRAL20	PFASUM45	14	1	0.1658
ASTRAL20	PFASUM46	15	1	0.1682
ASTRAL20	PFASUM47	15	1	0.1695
ASTRAL20	PFASUM48	15	1	0.1701
ASTRAL20	PFASUM49	17	1	0.1673
ASTRAL20	PFASUM50	17	1	0.1666
ASTRAL20	PFASUM51	15	1	0.1672
ASTRAL20	PFASUM52	15	1	0.1686
ASTRAL20	PFASUM53	15	1	0.1670
ASTRAL20	PFASUM54	15	1	0.1671
ASTRAL20	PFASUM55	15	1	0.1661

Subset	Matrix	Gop	Ext	Cov.
ASTRAL20	PFASUM56	12	2	0.1655
ASTRAL20	PFASUM57	15	1	0.1647
ASTRAL20	PFASUM58	16	1	0.1674
ASTRAL20	PFASUM59	16	1	0.1688
ASTRAL20	PFASUM60	16	1	0.1706
ASTRAL20	PFASUM61	12	2	0.1661
ASTRAL20	PFASUM62	16	1	0.1653
ASTRAL20	PFASUM63	15	1	0.1629
ASTRAL20	PFASUM64	15	1	0.1627
ASTRAL20	PFASUM65	16	1	0.1606
ASTRAL20	PFASUM66	11	2	0.1596
ASTRAL20	PFASUM67	15	1	0.1587
ASTRAL20	PFASUM68	15	1	0.1544
ASTRAL20	PFASUM69	16	1	0.1566
ASTRAL20	PFASUM70	16	1	0.1566
ASTRAL20	PFASUM71	18	1	0.1559
ASTRAL20	PFASUM72	17	1	0.1550
ASTRAL20	PFASUM73	15	1	0.1524
ASTRAL20	PFASUM74	9	1	0.1542
ASTRAL20	PFASUM75	9	1	0.1538
ASTRAL20	PFASUM76	9	1	0.1523
ASTRAL20	PFASUM77	9	1	0.1531
ASTRAL20	PFASUM78	9	1	0.1546
ASTRAL20	PFASUM79	9	1	0.1531
ASTRAL20	PFASUM80	10	1	0.1493
ASTRAL20	PFASUM81	9	1	0.1445
ASTRAL20	PFASUM82	9	1	0.1443
ASTRAL20	PFASUM83	9	1	0.1443
ASTRAL20	PFASUM84	9	1	0.1443
ASTRAL20	PFASUM85	10	1	0.1440
ASTRAL20	PFASUM86	10	1	0.1440
ASTRAL20	PFASUM87	10	1	0.1440
ASTRAL20	PFASUM88	10	1	0.1440
ASTRAL20	PFASUM89	10	1	0.1440
ASTRAL20	PFASUM90	9	1	0.1453
ASTRAL20	PFASUM91	9	1	0.1453
ASTRAL20	PFASUM92	9	1	0.1453
ASTRAL20	PFASUM93	9	1	0.1453
ASTRAL20	PFASUM94	9	1	0.1453
ASTRAL20	PFASUM95	9	1	0.1453
ASTRAL20	PFASUM96	9	1	0.1453
ASTRAL20	PFASUM97	9	1	0.1453
ASTRAL20	PFASUM98	9	1	0.1453
ASTRAL20	PFASUM99	9	1	0.1453
ASTRAL20	PFASUM100	10	1	0.1495

Table C.5: Highest achieved coverage values (Cov.) on the ASTRAL20 subset (version 2.06) obtained for *PFASUM Search Matrices* and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Matrix	Gop	Ext	Cov.
ASTRAL40	PFASUM11	20	2	0.1992
ASTRAL40	PFASUM12	20	3	0.3137
ASTRAL40	PFASUM13	20	2	0.3902
ASTRAL40	PFASUM14	20	2	0.4071
ASTRAL40	PFASUM15	18	2	0.4153
ASTRAL40	PFASUM16	16	3	0.4151
ASTRAL40	PFASUM17	20	2	0.4197
ASTRAL40	PFASUM18	20	2	0.4205
ASTRAL40	PFASUM19	17	2	0.4257
ASTRAL40	PFASUM20	16	2	0.4274
ASTRAL40	PFASUM21	15	2	0.4306
ASTRAL40	PFASUM22	19	1	0.4323
ASTRAL40	PFASUM23	15	2	0.4341
ASTRAL40	PFASUM24	17	2	0.4335
ASTRAL40	PFASUM25	17	2	0.4346
ASTRAL40	PFASUM26	16	2	0.4373
ASTRAL40	PFASUM27	17	2	0.4389
ASTRAL40	PFASUM28	17	1	0.4365
ASTRAL40	PFASUM29	17	1	0.4401
ASTRAL40	PFASUM30	17	1	0.4406
ASTRAL40	PFASUM31	17	1	0.4427
ASTRAL40	PFASUM32	17	1	0.4422
ASTRAL40	PFASUM33	18	1	0.4427
ASTRAL40	PFASUM34	17	1	0.4430
ASTRAL40	PFASUM35	17	1	0.4425
ASTRAL40	PFASUM36	17	1	0.4428
ASTRAL40	PFASUM37	17	1	0.4429
ASTRAL40	PFASUM38	18	1	0.4425
ASTRAL40	PFASUM39	15	2	0.4413
ASTRAL40	PFASUM40	19	1	0.4437
ASTRAL40	PFASUM41	14	2	0.4447
ASTRAL40	PFASUM42	14	2	0.4437
ASTRAL40	PFASUM43	13	1	0.4448
ASTRAL40	PFASUM44	13	1	0.4428
ASTRAL40	PFASUM45	13	1	0.4448
ASTRAL40	PFASUM46	14	1	0.4420
ASTRAL40	PFASUM47	11	2	0.4415
ASTRAL40	PFASUM48	16	1	0.4425
ASTRAL40	PFASUM49	15	1	0.4426
ASTRAL40	PFASUM50	16	1	0.4433
ASTRAL40	PFASUM51	17	1	0.4418
ASTRAL40	PFASUM52	15	1	0.4411
ASTRAL40	PFASUM53	11	2	0.4426
ASTRAL40	PFASUM54	16	1	0.4419
ASTRAL40	PFASUM55	11	2	0.4430
ASTRAL40	PFASUM56	11	2	0.4429
ASTRAL40	PFASUM57	11	2	0.4445
ASTRAL40	PFASUM58	15	1	0.4411
ASTRAL40	PFASUM59	11	2	0.4416
ASTRAL40	PFASUM60	15	1	0.4412
ASTRAL40	PFASUM61	11	2	0.4392
ASTRAL40	PFASUM62	11	2	0.4394
ASTRAL40	PFASUM63	15	1	0.4391
ASTRAL40	PFASUM64	12	2	0.4390
ASTRAL40	PFASUM65	14	1	0.4396
ASTRAL40	PFASUM66	11	2	0.4390
ASTRAL40	PFASUM67	9	3	0.4388
ASTRAL40	PFASUM68	14	1	0.4395
ASTRAL40	PFASUM69	14	1	0.4384
ASTRAL40	PFASUM70	14	1	0.4402
ASTRAL40	PFASUM71	16	1	0.4362
ASTRAL40	PFASUM72	16	1	0.4386
ASTRAL40	PFASUM73	16	1	0.4386
ASTRAL40	PFASUM74	9	1	0.4380
ASTRAL40	PFASUM75	9	1	0.4390
ASTRAL40	PFASUM76	9	1	0.4380
ASTRAL40	PFASUM77	10	1	0.4370
ASTRAL40	PFASUM78	10	1	0.4372
ASTRAL40	PFASUM79	10	1	0.4368
ASTRAL40	PFASUM80	10	1	0.4360
ASTRAL40	PFASUM81	10	1	0.4341
ASTRAL40	PFASUM82	10	1	0.4342
ASTRAL40	PFASUM83	10	1	0.4342
ASTRAL40	PFASUM84	10	1	0.4342
ASTRAL40	PFASUM85	9	1	0.4340
ASTRAL40	PFASUM86	9	1	0.4340
ASTRAL40	PFASUM87	9	1	0.4340
ASTRAL40	PFASUM88	9	1	0.4340
ASTRAL40	PFASUM89	9	1	0.4340
ASTRAL40	PFASUM90	9	1	0.4346
ASTRAL40	PFASUM91	9	1	0.4346
ASTRAL40	PFASUM92	9	1	0.4346
ASTRAL40	PFASUM93	9	1	0.4346
ASTRAL40	PFASUM94	9	1	0.4346
ASTRAL40	PFASUM95	9	1	0.4346
ASTRAL40	PFASUM96	9	1	0.4346
ASTRAL40	PFASUM97	9	1	0.4346
ASTRAL40	PFASUM98	9	1	0.4346
ASTRAL40	PFASUM99	9	1	0.4346
ASTRAL40	PFASUM100	10	1	0.4373

Table C.6: Highest achieved coverage values (Cov.) on the ASTRAL40 subset (version 2.06) obtained for *PFASUM Search Matrices* and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

Subset	Matrix	Gop	Ext	Cov.
ASTRAL70	PFASUM11	19	2	0.3406
ASTRAL70	PFASUM12	20	3	0.4506
ASTRAL70	PFASUM13	16	3	0.5125
ASTRAL70	PFASUM14	20	2	0.5252
ASTRAL70	PFASUM15	16	3	0.5316
ASTRAL70	PFASUM16	18	2	0.5328
ASTRAL70	PFASUM17	20	2	0.5354
ASTRAL70	PFASUM18	18	2	0.5361
ASTRAL70	PFASUM19	17	2	0.5397
ASTRAL70	PFASUM20	16	2	0.5404
ASTRAL70	PFASUM21	15	2	0.5434
ASTRAL70	PFASUM22	15	2	0.5455
ASTRAL70	PFASUM23	15	2	0.5458
ASTRAL70	PFASUM24	17	2	0.5444
ASTRAL70	PFASUM25	17	2	0.5453
ASTRAL70	PFASUM26	17	2	0.5457
ASTRAL70	PFASUM27	17	2	0.5473
ASTRAL70	PFASUM28	13	2	0.5465
ASTRAL70	PFASUM29	13	2	0.5481
ASTRAL70	PFASUM30	13	2	0.5502
ASTRAL70	PFASUM31	13	2	0.5508
ASTRAL70	PFASUM32	17	1	0.5499
ASTRAL70	PFASUM33	17	1	0.5499
ASTRAL70	PFASUM34	17	1	0.5500
ASTRAL70	PFASUM35	12	2	0.5496
ASTRAL70	PFASUM36	17	1	0.5496
ASTRAL70	PFASUM37	14	2	0.5483
ASTRAL70	PFASUM38	14	2	0.5490
ASTRAL70	PFASUM39	14	2	0.5475
ASTRAL70	PFASUM40	19	1	0.5494
ASTRAL70	PFASUM41	15	2	0.5502
ASTRAL70	PFASUM42	18	1	0.5493
ASTRAL70	PFASUM43	12	1	0.5479
ASTRAL70	PFASUM44	12	1	0.5487
ASTRAL70	PFASUM45	10	2	0.5484
ASTRAL70	PFASUM46	13	1	0.5475
ASTRAL70	PFASUM47	11	2	0.5496
ASTRAL70	PFASUM48	10	2	0.5492
ASTRAL70	PFASUM49	10	2	0.5497
ASTRAL70	PFASUM50	10	2	0.5484
ASTRAL70	PFASUM51	10	2	0.5460
ASTRAL70	PFASUM52	10	2	0.5472
ASTRAL70	PFASUM53	10	2	0.5472
ASTRAL70	PFASUM54	10	2	0.5476
ASTRAL70	PFASUM55	10	2	0.5470
ASTRAL70	PFASUM56	10	2	0.5467
ASTRAL70	PFASUM57	10	2	0.5475
ASTRAL70	PFASUM58	10	2	0.5463
ASTRAL70	PFASUM59	14	1	0.5449
ASTRAL70	PFASUM60	11	2	0.5448
ASTRAL70	PFASUM61	14	1	0.5445
ASTRAL70	PFASUM62	10	2	0.5440
ASTRAL70	PFASUM63	10	2	0.5443
ASTRAL70	PFASUM64	10	2	0.5445
ASTRAL70	PFASUM65	14	1	0.5441
ASTRAL70	PFASUM66	10	2	0.5433
ASTRAL70	PFASUM67	10	2	0.5433
ASTRAL70	PFASUM68	14	1	0.5429
ASTRAL70	PFASUM69	11	2	0.5426
ASTRAL70	PFASUM70	15	1	0.5432
ASTRAL70	PFASUM71	11	2	0.5423
ASTRAL70	PFASUM72	11	2	0.5430
ASTRAL70	PFASUM73	11	2	0.5427
ASTRAL70	PFASUM74	8	1	0.5415
ASTRAL70	PFASUM75	9	1	0.5426
ASTRAL70	PFASUM76	8	1	0.5422
ASTRAL70	PFASUM77	8	1	0.5421
ASTRAL70	PFASUM78	9	1	0.5433
ASTRAL70	PFASUM79	9	1	0.5436
ASTRAL70	PFASUM80	8	1	0.5397
ASTRAL70	PFASUM81	8	1	0.5401
ASTRAL70	PFASUM82	8	1	0.5399
ASTRAL70	PFASUM83	8	1	0.5399
ASTRAL70	PFASUM84	8	1	0.5399
ASTRAL70	PFASUM85	8	1	0.5399
ASTRAL70	PFASUM86	8	1	0.5399
ASTRAL70	PFASUM87	8	1	0.5399
ASTRAL70	PFASUM88	8	1	0.5399
ASTRAL70	PFASUM89	8	1	0.5399
ASTRAL70	PFASUM90	8	1	0.5401
ASTRAL70	PFASUM91	8	1	0.5401
ASTRAL70	PFASUM92	8	1	0.5401
ASTRAL70	PFASUM93	8	1	0.5401
ASTRAL70	PFASUM94	8	1	0.5401
ASTRAL70	PFASUM95	8	1	0.5401
ASTRAL70	PFASUM96	8	1	0.5401
ASTRAL70	PFASUM97	8	1	0.5401
ASTRAL70	PFASUM98	8	1	0.5401
ASTRAL70	PFASUM99	8	1	0.5401
ASTRAL70	PFASUM100	9	1	0.5417

Table C.7: Highest achieved coverage values (Cov.) on the ASTRAL70 subset (version 2.06) obtained for *PFASUM Search Matrices* and the corresponding gap opening (Gop) and extension (Ext) penalty parameters.

	PFASUM31 entropy level					PFASUM43 entropy level					PFASUM60 entropy level				
	Matrix	**Gop**	**Ext**	**Cov.**	***Z*-score**	**Matrix**	**Gop**	**Ext**	**Cov.**	***Z*-score**	**Matrix**	**Gop**	**Ext**	**Cov.**	***Z*-score**
ASTRAL20	PFASUM31	15	2	0.1562		PFASUM43	14	1	0.1650		PFASUM60	16	1	0.1706	
	BLOSUM37	13	3	0.1288	-168.7982	BLOSUM43	14	1	0.1498	-92.3872	BLOSUM51	15	1	0.1493	-122.8447
	CorBLOSUM35	20	1	0.1409	-91.9623	CorBLOSUM41	14	1	0.1491	-94.2003	CorBLOSUM50	14	1	0.1515	-112.3579
	PAM316	8	2	0.0934	-410.7383	PAM258	11	3	0.1055	-365.2371	PAM203	12	3	0.1155	-324.8256
	RBLOSUM37	14	3	0.1402	-96.4736	RBLOSUM43	14	1	0.1494	-93.8165	RBLOSUM52	16	1	0.1544	-92.4120
	VTML270	11	2	0.1433	-75.1459	VTML226	15	1	0.1583	-36.1578	VTML182	16	1	0.1591	-65.5892
ASTRAL40	PFASUM31	17	1	0.4427		PFASUM43	13	1	0.4448		PFASUM60	15	1	0.4412	
	BLOSUM37	15	1	0.4254	-91.3180	BLOSUM43	13	1	0.4335	-60.2968	BLOSUM51	15	1	0.4356	-28.6005
	CorBLOSUM35	15	1	0.4344	-42.1861	CorBLOSUM41	12	1	0.4318	-68.8085	CorBLOSUM50	15	1	0.4377	-17.4336
	PAM316	9	2	0.3833	-307.7493	PAM258	12	1	0.3971	-249.5805	PAM203	18	1	0.4052	-190.3504
	RBLOSUM37	16	1	0.4351	-39.2649	RBLOSUM43	12	1	0.4343	-54.2841	RBLOSUM52	15	1	0.4375	-17.3178
	VTML270	12	1	0.4277	-77.3459	VTML226	13	1	0.4366	-42.6708	VTML182	14	1	0.4415	**4.2405**
ASTRAL70	PFASUM31	13	2	0.5508		PFASUM43	12	1	0.5479		PFASUM60	11	2	0.5448	
	BLOSUM37	13	2	0.5349	-91.4920	BLOSUM43	10	2	0.5423	-33.3238	BLOSUM51	15	1	0.5414	-20.8834
	CorBLOSUM35	17	1	0.5403	-59.6804	CorBLOSUM41	13	1	0.5418	-35.5896	CorBLOSUM50	11	2	0.5420	-16.3138
	PAM316	11	1	0.5008	-285.8505	PAM258	11	2	0.5134	-207.1712	PAM203	17	1	0.5215	-134.4453
	RBLOSUM37	13	2	0.5413	-55.6744	RBLOSUM43	12	1	0.5440	-23.0401	RBLOSUM52	12	1	0.5420	-16.8654
	VTML270	9	3	0.5397	-64.3520	VTML226	12	2	0.5444	-20.0736	VTML182	10	2	0.5466	**10.8274**

Table C.8: Highest achieved coverage scores (Cov.) for the comparison of PFASUM31, PFASUM43, and PFASUM60 with their (R/Cor)BLOSUM, VTML, and PAM counterparts based on similar relative matrix entropy and the corresponding optimal gap opening (Gop) and extension (Ext) penalties for each matrix. Also shown is the computed *Z*-score for the comparison between a particular PFASUM matrix and its counterpart based on 500 bootstrap rounds. Negative *Z*-scores indicate a performance advantage of *PFASUM Search Matrices* over their counterparts. A positive *Z*-score (red) indicates a lower performance of *PFASUM Search Matrices* compared to its counterpart.

Subset	Matrix	Gop	Ext	Cov.	Z-score
ASTRAL20	BLOSUM50	15	1	0.1474	-124.9418
ASTRAL20	PFASUM59	16	1	0.1688	
ASTRAL20	BLOSUM62	10	1	0.1465	-48.0527
ASTRAL20	PFASUM78	9	1	0.1546	
ASTRAL20	PAM250	9	3	0.1105	-348.2035
ASTRAL20	PFASUM45	14	1	0.1658	
ASTRAL20	VTML160	16	1	0.1566	-12.0228
ASTRAL20	PFASUM67	15	1	0.1587	
ASTRAL20	VTML200	14	1	0.1598	-40.5518
ASTRAL20	PFASUM51	15	1	0.1672	
ASTRAL40	BLOSUM50	11	2	0.4371	-22.6776
ASTRAL40	PFASUM59	11	2	0.4416	
ASTRAL40	BLOSUM62	10	1	0.4346	-10.4120
ASTRAL40	PFASUM78	10	1	0.4372	
ASTRAL40	PAM250	10	2	0.4024	-216.6223
ASTRAL40	PFASUM45	13	1	0.4448	
ASTRAL40	VTML160	15	1	0.4386	**-0.0428**
ASTRAL40	PFASUM67	9	3	0.4388	
ASTRAL40	VTML200	14	1	0.4392	-12.1233
ASTRAL40	PFASUM51	17	1	0.4418	
ASTRAL70	BLOSUM50	15	1	0.5420	-16.6450
ASTRAL70	PFASUM59	14	1	0.5449	
ASTRAL70	BLOSUM62	9	1	0.5402	-18.8916
ASTRAL70	PFASUM78	9	1	0.5433	
ASTRAL70	PAM250	11	2	0.5187	-172.2656
ASTRAL70	PFASUM45	10	2	0.5484	
ASTRAL70	VTML160	11	2	0.5448	**8.4916**
ASTRAL70	PFASUM67	10	2	0.5433	
ASTRAL70	VTML200	9	2	0.5459	**1.2365**
ASTRAL70	PFASUM51	10	2	0.5460	

Table C.9: Highest achieved coverage scores (Cov.) for the comparison of *Standard Search Matrices* and *PFASUM Search Matrices* based on similar relative matrix entropy and the corresponding optimal gap opening (Gop) and extension (Ext) penalties for each matrix. Also shown is the computed Z-score for each individual comparison based on 500 bootstrap rounds. Negative Z-scores indicate a performance advantage of *PFASUM Search Matrices* over *Standard Search Matrices*. Positive Z-scores (blue) refer to *Standard Search Matrices* that perform better than their PFASUM counterpart. Insignificant performance differences are highlighted in red.

ASTRAL20				ASTRAL40				ASTRAL70			
Matrix	**Gop**	**Ext**	**Cov.**	**Matrix**	**Gop**	**Ext**	**Cov.**	**Matrix**	**Gop**	**Ext**	**Cov.**
PFASUM31	15	2	0.1562	PFASUM31	17	1	0.4427	PFASUM31	13	2	0.5508
PFASUM43	14	1	0.165	PFASUM43	13	1	0.4448	PFASUM43	12	1	0.5479
PFASUM60	16	1	0.1706	PFASUM60	15	1	0.4412	PFASUM60	11	2	0.5448
BLOSUM50	15	1	0.1474	BLOSUM50	11	2	0.4371	BLOSUM50	15	1	0.542
BLOSUM62	10	1	0.1465	BLOSUM62	10	1	0.4346	BLOSUM62	9	1	0.5402
BLOSUM80	8	1	0.1231	BLOSUM80	9	1	0.414	BLOSUM80	8	1	0.5256
MD10	6	2	0.0206	MD10	7	2	0.1923	MD10	6	2	0.3494
MD20	6	2	0.032	MD20	7	2	0.239	MD20	7	2	0.3889
MD40	8	2	0.0512	MD40	9	2	0.2996	MD40	9	2	0.4381
OPTIMA5	19	3	0.1563	OPTIMA5	14	3	0.4383	OPTIMA5	17	3	0.5442
PAM120	9	1	0.0988	PAM120	9	1	0.3976	PAM120	10	1	0.5092
PAM250	9	3	0.1105	PAM250	10	2	0.4024	PAM250	11	2	0.5187
VTML10	5	1	0.0192	VTML10	6	1	0.1909	VTML10	5	1	0.3476
VTML20	5	1	0.0331	VTML20	6	1	0.2468	VTML20	6	1	0.3937
VTML40	7	1	0.0557	VTML40	7	1	0.3162	VTML40	7	1	0.4472
VTML80	8	1	0.0951	VTML80	9	1	0.389	VTML80	8	1	0.5044
VTML120	9	1	0.131	VTML120	9	1	0.426	VTML120	9	1	0.5337
VTML160	16	1	0.1566	VTML160	15	1	0.4386	VTML160	11	2	0.5448
VTML200	14	1	0.1598	VTML200	14	1	0.4392	VTML200	9	2	0.5459

Table C.10: Highest achieved coverage scores (Cov.) for the comparison of PFASUM31, PFASUM43, and PFASUM60 with *Standard Search Matrices* and the corresponding optimal gap opening (Gop) and extension (Ext) penalties for each matrix.

Subset	PFASUM	BLOSUM50	BLOSUM62	BLOSUM80	MD10	MD20	MD40	OPTIMA5	PAM120	PAM250	VTML10
ASTRAL20	PFASUM31	51.26	57.54	203.63	1061.17	955.94	757.71	**-1.70**	363.50	294.64	1070.21
	PFASUM43	101.83	108.95	256.11	1116.56	1011.97	813.08	50.27	416.42	348.98	1125.43
	PFASUM60	129.70	137.01	280.25	1098.78	1000.60	814.22	80.21	435.32	370.41	1107.28
ASTRAL40	PFASUM31	29.28	39.56	145.18	1378.02	1097.67	737.82	22.12	228.64	207.94	1363.20
	PFASUM43	40.45	50.85	158.76	1425.60	1136.37	765.42	33.09	244.26	223.43	1408.99
	PFASUM60	21.09	31.86	142.02	1440.62	1142.94	762.01	13.64	229.21	207.73	1422.68
ASTRAL70	PFASUM31	49.87	62.02	143.79	1125.31	905.95	648.85	37.38	229.92	182.64	1183.97
	PFASUM43	33.53	45.96	130.36	1138.22	913.05	649.60	20.79	219.02	170.28	1200.51
	PFASUM60	15.50	27.78	112.45	1118.18	893.55	630.14	2.91	201.42	152.24	1179.30

Subset	PFASUM	VTML20	VTML40	VTML80	VTML120	VTML160	VTML200	PFASUM31	PFASUM43	PFASUM60
ASTRAL20	PFASUM31	944.53	706.49	386.25	148.43	-2.74	-20.92	0.00	-51.14	-80.63
	PFASUM43	1000.55	761.17	438.81	199.51	49.05	31.72	51.14	0.00	-30.83
	PFASUM60	989.95	765.40	456.91	224.90	78.94	62.52	80.63	30.83	0.00
ASTRAL40	PFASUM31	1024.10	638.93	271.24	83.25	20.86	17.50	0.00	-10.51	9.44
	PFASUM43	1059.21	663.47	287.76	95.73	32.21	28.72	10.51	0.00	20.64
	PFASUM60	1062.97	657.55	273.66	77.38	12.05	8.58	-9.44	-20.64	0.00
ASTRAL70	PFASUM31	854.69	572.76	272.88	98.63	34.94	26.75	0.00	17.42	34.96
	PFASUM43	859.63	570.82	262.94	83.79	18.69	10.18	-17.42	0.00	18.10
	PFASUM60	840.87	552.36	244.00	65.75	**1.29**	-7.33	-34.96	-18.10	0.00

Table C.11: Table of Z-score values for the comparison between *PFASUM Search Matrices* and *Standard Search Matrices* on the three different ASTRAL datasets. Z-scores with $|Z| \geq 1.96$ represent statistically significant underlying distributions at the 95% confidence interval. Non-significant Z-scores are highlight in red.

	Matrix	average $\mathcal{Q}$-score
Twilight Zone	BLOSUM50	0.3913
	BLOSUM62	0.3914
	PAM250	0.3710
	PFASUM31	0.4034
	PFASUM43	0.4002
	PFASUM60	**0.4048**
	VTML160	0.3997
	VTML200	0.3885
Superfamilies	BLOSUM50	0.6342
	BLOSUM62	0.6343
	PAM250	0.6115
	PFASUM31	**0.6417**
	PFASUM43	0.6378
	PFASUM60	0.6407
	VTML160	0.6379
	VTML200	0.6381

Table C.12: Average q-score $\bar{q}$ for the SABmark alignments, split between superfamily alignments and so-called "twilight zone" alignments. Matrices with the highest performances are highlighted in bold.

Appendix D
(Co-)Authored publications

- **PFASUM: A substitution matrix from Pfam structural alignments**
  *Frank Keul, **Martin Hess**, Michael Goesele and K. Hamacher*
  BMC Bioinformatics, Vol. 18, pp. 293, 2017
- **Visual Analysis and Comparison of Multiple Sequence Alignments**
  ***Martin Hess**, Daniel Jente, Josef Wiemeyer, Kay Hamacher and Michael Goesele*
  In: 6th Eurographics Workshop on Visual Computing for Biology and Medicine, 2016
- **Addressing inaccuracies in BLOSUM computation improves homology search performance.**
  ***Martin Hess**, Frank Keul, Michael Goesele and K. Hamacher*
  BMC Bioinformatics, Vol. 17, pp. 189, 2016
- **Serious Games for Solving Protein Sequence Alignments-Combining Citizen Science and Gaming**
  ***Martin Hess**, Josef Wiemeyer, Kay Hamacher and Michael Goesele*
  In: Games for Training, Education, Health and Sports. 4th International Conference on Serious Games, GameDays 2014
- **Visual exploration of parameter influence on phylogenetic trees**
  ***Martin Hess**, Sebastian Bremm, Stephanie Weissgraeber, Kay Hamacher, Michael Goesele, Josef Wiemeyer and Tatiana von Landesberger*
  IEEE Computer Graphics and Applications, Special Issue – BioVis, Vol. 34, No. 2, pp. 48-56, 2014
- **PCDC – On the Highway to Data – A Tool for the Fast Generation of Large Synthetic Data Sets**
  *Sebastian Bremm, **Martin Hess**, Tatiana von Landesberger and Dieter W. Fellner*
  In: International Workshop on Visual Analytics, 2012
- **Interactive Visual Comparison of Multiple Trees**
  *Sebastian Bremm, Tatiana von Landesberger, **Martin Heß**, Tobias Schreck, Philipp Weil and Kay Hamacher*
  IEEE Visual Analytics Science and Technology, 2011

Appendix E
Curriculum Vitae

Martin Philipp Heß

2013 - 2017	Research Associate at Technische Universität Darmstadt, GCC - Graphics, Capture and Massively Parallel Computing
2013	Master of Science in Computer Science & Master of Science in Visual Computing Thesis title: “Visual-interactive Comparison of Many Hierarchical Structures” Referee: Prof. Dr. Dieter W. Fellner
2010	Bachelor of Science in Computer Science Thesis title: “Visual Comparison of Hierarchically Organized Data” Referee: Prof. Dr. Dieter W. Fellner
2003 - 2013	Study of computer science and visual computing at Technische Universität Darmstadt

Bibliography

[Agrawal and Huang 2009] Agrawal, A. and X. Huang (2009). "Pairwise statistical significance of local sequence alignment using multiple parameter sets and empirical justification of parameter set change penalty". In: *BMC Bioinformatics* 10.3, S1. DOI: 10.1186/1471-2105-10-S3-S1 (cited on pages 6, 14, 123).

[Altschul 1991] Altschul, S. F. (1991). "Amino acid substitution matrices from an information theoretic perspective". In: *Journal of Molecular Biology* 219.3, pp. 555–565 (cited on pages 85, 90, 92, 106, 110, 116).

[Altschul and Gish 1996] Altschul, S. F. and W. Gish (1996). "Local alignment statistics". In: *Methods in Enzymology* 266, pp. 460 –480. ISSN: 0076-6879. DOI: 10.1016/S0076-6879(96)66029-7 (cited on page 21).

[Altschul et al. 1990] Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman (1990). "Basic local alignment search tool". In: *Journal of Molecular Biology* 215.3, pp. 403–410. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2 (cited on pages 5, 14, 21–23, 67, 76, 79, 80, 92, 106, 117).

[Altschul et al. 1997] Altschul, S. F., T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". In: *Nucleic Acids Research* 25.17, pp. 3389–3402. DOI: 10.1093/nar/25.17.3389 (cited on pages 5, 21, 23).

[Anderson et al. 2011] Anderson, C. L., C. L. Strope, and E. N. Moriyama (2011). "SuiteMSA: visual tools for multiple sequence alignment comparison and molecular sequence simulation". In: *BMC Bioinformatics* 12.1. ISSN: 1471-2105 (cited on pages 124, 126, 128, 142, 147).

[Angermüller et al. 2012] Angermüller, C., A. Biegert, and J. Söding (2012). "Discriminative modelling of context-specific amino acid substitution probabilities". In: *Bioinformatics* 28.24, pp. 3240–3247 (cited on page 92).

[Benner et al. 1993] Benner, S. A., M. A. Cohen, and G. H. Gonnet (1993). "Empirical and Structural Models for Insertions and Deletions in the Divergent Evolution of Proteins". In: *Journal of Molecular Biology* 229.4, pp. 1065 –1082. DOI: 10.1006/jmbi.1993.1105 (cited on page 14).

[Bitkom Research 2015] Bitkom Research (2015). *Gaming Monitor Deutschland 2015*. Tech. rep. Bitkom Research GmbH (cited on page 41).

[Blackshields et al. 2010] Blackshields, G., F. Sievers, W. Shi, A. Wilm, and D. G. Higgins (2010). "Sequence embedding for fast construction of guide trees for multiple sequence alignment". In: *Algorithms for Molecular Biology* 5.1, p. 21 (cited on page 37).

[Boratyn et al. 2012] Boratyn, G. M., A. A. Schäffer, R. Agarwala, S. F. Altschul, D. J. Lipman, and T. L. Madden (2012). "Domain enhanced lookup time accelerated BLAST". In: *Biology Direct* 7.1, p. 12. ISSN: 1745-6150. DOI: 10.1186/1745-6150-7-12 (cited on page 23).

[Boutonnet et al. 1995] Boutonnet, N. S., M. J. Rooman, M.-E. Ochagavia, J. Richelle, and S. J. Wodak (1995). "Optimal protein structure alignments by multiple linkage clustering: application to distantly related proteins". In: *Protein Engineering* 8.7, pp. 647–662. DOI: 10.1093/protein/8.7.647 (cited on pages 39, 118).

[Brenner et al. 1998] Brenner, S. E., C. Chothia, and T. J. Hubbard (1998). "Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships". In: *Proceedings of the National Academy of Sciences* 95.11, pp. 6073–6078 (cited on pages 79, 80, 107).

[Brenner et al. 2000] Brenner, S. E., P. Koehl, and M. Levitt (2000). "The ASTRAL compendium for protein structure and sequence analysis". In: *Nucleic Acids Research* 28.1, pp. 254–256 (cited on pages 79, 106).

[Carrillo and Lipman 1988] Carrillo, H. and D. Lipman (1988). "The multiple sequence alignment problem in biology". In: *SIAM Journal on Applied Mathematics* 48.5, pp. 1073–1082 (cited on pages 28, 29).

[Chandonia et al. 2004] Chandonia, J.-M., G. Hon, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S. E. Brenner (2004). "The ASTRAL Compendium in 2004." In: *Nucleic Acids Research* 32.Database issue, pp. D189–D192. DOI: 10.1093/nar/gkh034 (cited on pages 79, 106).

[Chatzou et al. 2016] Chatzou, M., C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame (2016). "Multiple sequence alignment modeling: methods and applications". In: *Briefings in Bioinformatics* 17.6, pp. 1009–1023. DOI: 10.1093/bib/bbv099 (cited on pages 3, 5, 26, 29, 32, 33, 79, 123, 124, 144, 147).

[Cline et al. 2002] Cline, M., R. Hughey, and K. Karplus (2002). "Predicting reliable regions in protein sequence alignments". In: *Bioinformatics* 18.2 (cited on pages 40, 124, 125, 131).

[Collins et al. 1988] Collins, J., A. Coulson, and A Lyall (1988). "The significance of protein sequence similarities". In: *Bioinformatics* 4.1, pp. 67–71 (cited on page 21).

[Cooper et al. 2010a] Cooper, S., F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, et al. (2010a). "Predicting protein structures with a multiplayer online game". In: *Nature* 466.7307, pp. 756–760. DOI: 10.1038/nature09304 (cited on page 45).

[Cooper et al. 2010b] Cooper, S., A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, et al. (2010b). "The challenge of designing scientific discovery games". In: *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. ACM, pp. 40–47 (cited on page 41).

[Cooper et al. 2011] Cooper, S., F. Khatib, I. Makedon, H. Lu, J. Barbero, D. Baker, J. Fogarty, Z. Popović, and F. players (2011). "Analysis of social gameplay macros in the Foldit cookbook". In: *Proceedings of the 6th International Conference on Foundations of Digital Games*. FDG '11. Bordeaux, France: ACM, pp. 9–14. ISBN: 978-1-4503-0804-5. DOI: 10.1145/2159365.2159367 (cited on page 46).

[Csikszentmihalyi 2000] Csikszentmihalyi, M. (2000). *Beyond boredom and anxiety*. San Francisco, CA, US: Jossey-Bass (cited on page 53).

[Dayhoff et al. 1978] Dayhoff, M. O., R. M. Schwartz, and B. C. Orcutt (1978). "A model of evolutionary change in proteins". In: *Atlas of Protein Sequence and Structure*. Ed. by M. O. Dayhoff. Vol. 5. Washington, DC: Natl. Biomed. Res. Found., pp. 345–352 (cited on pages 4, 6, 76).

[Do et al. 2005] Do, C. B., M. S. Mahabhashyam, M. Brudno, and S. Batzoglou (2005). "ProbCons: Probabilistic consistency-based multiple sequence alignment". In: *Genome Research* 15.2, pp. 330–340. DOI: 10.1101/gr.2821705 (cited on pages 31, 32).

[Dörner et al. 2016] Dörner, R., S. Göbel, W. Effelsberg, and J. Wiemeyer (2016). *Serious Games: Foundations, Concepts and Practice*. Springer International Publishing. DOI: 10.1007/978-3-319-40612-1 (cited on pages 41, 42).

[ESA 2017] ESA (2017). *2017 Essential Facts About the Computer and Video Game Industry*. Tech. rep. Entertainment Software Association (cited on page 41).

[Eddy 1998] Eddy, S. R. (1998). "Profile hidden Markov models." In: *Bioinformatics* 14.9, pp. 755–763. DOI: 10.1093/bioinformatics/14.9.755 (cited on page 37).

[Eddy 2009] Eddy, S. R. (2009). "A new generation of homology search tools based on probabilistic inference". In: *Genome Inform.* Vol. 23, pp. 205–211 (cited on page 14).

[Edgar 2009a] Edgar, R. (2009a). *MSA benchmark collection bench*. Accessed 22 Nov 2016. URL: http://www.drive5.com/bench/bench.tar.gz (cited on pages 39, 82, 117).

[Edgar 2009b] Edgar, R. (2009b). *qscore*. Accessed 22 Nov 2016. URL: http://www.drive5.com/qscore/qscore_src.tar.gz (cited on pages 82, 118).

[Edgar 2004a] Edgar, R. C. (2004a). "MUSCLE: a multiple sequence alignment method with reduced time and space complexity". In: *BMC Bioinformatics* 5.1, p. 113 (cited on pages 36, 118).

[Edgar 2004b] Edgar, R. C. (2004b). "MUSCLE: multiple sequence alignment with high accuracy and high throughput". In: *Nucleic Acids Research* 32.5, pp. 1792–1797. DOI: 10.1093/nar/gkh340 (cited on pages 6, 13, 14, 30, 31, 36, 37, 39, 67, 71, 77, 79, 117, 118, 125, 138).

[Edgar 2009c] Edgar, R. C. (2009c). "Optimizing substitution matrix choice and gap parameters for sequence alignment". In: *BMC Bioinformatics* 10.1. ISSN: 1471-2105 (cited on pages 118, 123).

[Edgar 2010] Edgar, R. C. (2010). “Quality measures for protein alignment benchmarks”. In: *Nucleic Acids Research* 38.7, pp. 2145–2153. DOI: 10.1093/nar/gkp1196 (cited on page 39).

[Elias 2006] Elias, I. (2006). “Settling the intractability of multiple alignment”. In: *Journal of Computational Biology* 13.7, pp. 1323–1339 (cited on pages 5, 6, 27, 29, 123).

[Eve Online 2017] Eve Online (2017). *Eve Online*. Accessed 17.05.2017. URL: https://www.eveonline.com (cited on page 44).

[Feng and Doolittle 1987] Feng, D.-F. and R. F. Doolittle (1987). “Progressive sequence alignment as a prerequisitetto correct phylogenetic trees”. In: *Journal of Molecular Evolution* 25.4, pp. 351–360 (cited on page 29).

[Finn et al. 2016] Finn, R. D. et al. (2016). “The Pfam protein families database: towards a more sustainable future”. In: *Nucleic Acids Research* 44.D1, pp. D279–D285. DOI: 10.1093/nar/gkv1344 (cited on pages 3, 4, 11, 16, 26, 63, 67, 101, 102, 121, 143, 168, 169).

[Fox et al. 2014] Fox, N. K., S. E. Brenner, and J.-M. Chandonia (2014). “SCOPe: Structural Classification of Proteins–extended, integrating SCOP and ASTRAL data and classification of new structures.” In: *Nucleic Acids Research* 42.Database issue, pp. D304–D309. DOI: 10.1093/nar/gkt1240 (cited on pages 79, 106).

[Garnier et al. 1996] Garnier, J., J.-F. Gibrat, and B. Robson (1996). “GOR method for predicting protein secondary structure from amino acid sequence”. In: *Computer Methods for Macromolecular Sequence Analysis*. Vol. 266. Methods in Enzymology Supplement C. Academic Press, pp. 540 –553. DOI: 10.1016/S0076-6879(96)66034-0 (cited on page 38).

[Gediga et al. 1999] Gediga, G., K.-C. Hamborg, and I. Düntsch (1999). “The IsoMetrics usability inventory: an operationalization of ISO 9241-10 supporting summative and formative evaluation of software systems”. In: *Behaviour & Information Technology* 18.3, pp. 151–164 (cited on page 50).

[Giribet and Wheeler 1999] Giribet, G. and W. C. Wheeler (1999). “On Gaps”. In: *Molecular Phylogenetics and Evolution* 13.1, pp. 132 –143. ISSN: 1055-7903. DOI: 10.1006/mpev.1999.0643 (cited on pages 6, 14, 123).

[Gonnet et al. 1992] Gonnet, G., M. Cohen, and S. Benner (1992). “Exhaustive matching of the entire protein sequence database”. In: *Science* 256.5062, pp. 1443–1445. ISSN: 0036-8075. DOI: 10.1126/science.1604319 (cited on page 76).

[Gotoh 1982] Gotoh, O. (1982). “An improved algorithm for matching biological sequences.” In: *Journal of Molecular Biology* 162.3, pp. 705–708 (cited on pages 13, 14, 18).

[Gouy et al. 2010] Gouy, M., S. Guindon, and O. Gascuel (2010). “SeaView Version 4: A Multiplatform Graphical User Interface for Sequence Alignment and Phylogenetic Tree Building”. In: *Molecular Biology and Evolution* 27.2 (cited on page 126).

[Green and Brenner 2002] Green, R. E. and S. E. Brenner (2002). “Bootstrapping and normalization for enhanced evaluations of pairwise sequence comparison”. In: *Proceedings of the IEEE* 90.12, pp. 1834–1847 (cited on pages 79–81, 93, 107).

[Gupta et al. 1995] Gupta, S. K., J. D. Kececioglu, and A. A. Schäffer (1995). “Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment”. In: *Journal of Computational Biology* 2.3, pp. 459–472 (cited on page 29).

[Hand 2010] Hand, E. (2010). “People power”. In: *Nature* 466.7307, pp. 685–687. DOI: 10.1038/466685a (cited on pages 41, 42).

[Henikoff and Henikoff 1991] Henikoff, S. and J. G. Henikoff (1991). “Automated assembly of protein blocks for database searching”. In: *Nucleic Acids Research* 19.23, pp. 6565–6572. DOI: 10.1093/nar/19.23.6565 (cited on pages 77, 83, 104).

[Henikoff and Henikoff 1992a] Henikoff, S. and J. G. Henikoff (1992a). “Amino acid substitution matrices from protein blocks”. In: *Proc Natl Acad Sci USA* 89.22, pp. 10915–10919 (cited on pages 4, 6, 8, 54, 60, 71, 76–78, 80, 83, 85, 87, 93, 103, 107, 125, 126, 132, 133, 144).

[Henikoff and Henikoff 1992b] Henikoff, S. and J. G. Henikoff (1992b). *BLOSUM source code*. Accessed 18.09.2015. URL: ftp://ftp.ncbi.nih.gov/repository/blocks/unix/blosum/blosum.tar.Z (cited on pages 8, 83, 84, 110).

[Henikoff and Henikoff 1993] Henikoff, S. and J. G. Henikoff (1993). “Performance evaluation of amino acid substitution matrices”. In: *Proteins: Struct, Funct, Bioinf* 17.1, pp. 49–61. ISSN: 1097-0134. DOI: 10.1002/prot.340170108 (cited on pages 77, 113).

[Hess et al. 2014a] Hess, M., S. Bremm, S. Weissgraeber, K. Hamacher, M. Goesele, J. Wiemeyer, and T. von Landesberger (2014a). “Visual Exploration of Parameter Influence on Phylogenetic Trees”. In: *IEEE Computer Graphics and Applications* 34.2. ISSN: 0272-1716 (cited on pages 6, 123, 137).

[Hess et al. 2014b] Hess, M., J. Wiemeyer, K. Hamacher, and M. Goesele (2014b). “Serious Games for Solving Protein Sequence Alignments - Combining Citizen Science and Gaming”. In: *Games for Training, Education, Health and Sports*. Springer International Publishing. ISBN: 978-3-319-05971-6. DOI: 10.1007/978-3-319-05972-3_18 (cited on pages 8, 47).

[Hess et al. 2016a] Hess, M., F. Keul, M. Goesele, and K. Hamacher (2016a). “Addressing inaccuracies in BLOSUM computation improves homology search performance”. In: *BMC Bioinformatics* 17.1, p. 189. ISSN: 1471-2105. DOI: 10.1186/s12859-016-1060-3 (cited on pages 4, 8, 76, 78, 79, 82, 83, 101, 102, 108, 113, 143, 145).

[Hess et al. 2016b] Hess, M., D. Jente, J. Wiemeyer, K. Hamacher, and M. Goesele (2016b). “Visual Analysis and Comparison of Multiple Sequence Alignments”. In: *Eurographics Workshop on Visual Computing for Biology and Medicine*. Ed. by S. Bruckner, B. Preim, A. Vilanova, H. Hauser, A. Hennemuth, and A. Lundervold. The Eurographics Association. ISBN: 978-3-03868-010-9. DOI: 10.2312/vcbm.20161268 (cited on pages 9, 143).

[Hirschberg 1975] Hirschberg, D. S. (1975). “A linear space algorithm for computing maximal common subsequences”. In: *Communications of the ACM* 18.6, 341–343. ISSN: 0001-0782. DOI: 10.1145/360825.360861 (cited on pages 14, 15).

[Hogeweg and Hesper 1984] Hogeweg, P. and B. Hesper (1984). “The alignment of sets of sequences and the construction of phyletic trees: an integrated method”. In: *Journal of Molecular Evolution* 20.2, pp. 175–186 (cited on page 29).

[Huang and Miller 1991] Huang, X. and W. Miller (1991). “A time-efficient, linear-space local similarity algorithm”. In: *Advances in Applied Mathematics* 12.3, pp. 337–357 (cited on page 35).

[IJsselsteijn et al. 2008] IJsselsteijn, W, K Poels, and Y. de Kort (2008). “The Game Experience Questionnaire: Development of a self-report measure to assess player experiences of digital games”. In: *TU Eindhoven, Eindhoven, The Netherlands* (cited on pages 42, 50, 56, 64, 150).

[Jones 1999] Jones, D. T. (1999). “Protein secondary structure prediction based on position-specific scoring matrices”. In: *Journal of Molecular Biology* 292.2 (cited on page 126).

[Jones et al. 1992] Jones, D. T., W. R. Taylor, and J. M. Thornton (1992). “The rapid generation of mutation data matrices from protein sequences”. In: *Computer Applications in the Biosciences* 8.3, pp. 275–282. DOI: 10.1093/bioinformatics/8.3.275 (cited on pages 76, 116).

[Just 2001] Just, W. (2001). “Computational complexity of multiple sequence alignment with SP-score”. In: *Journal of Computational Biology* 8.6, pp. 615–623 (cited on pages 5, 6, 27, 29, 123).

[Kann et al. 2000] Kann, M., B. Qian, and R. A. Goldstein (2000). “Optimization of a new score function for the detection of remote homologs”. In: *Proteins: Struct, Funct, Bioinf* 41.4, pp. 498–503. ISSN: 1097-0134. DOI: 10.1002/1097-0134(20001201)41:4<498::AID-PROT70>3.0.CO;2-3 (cited on page 76).

[Katoh et al. 2002] Katoh, K., K. Misawa, K.-i. Kuma, and T. Miyata (2002). “MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform”. In: *Nucleic Acids Research* 30.14, pp. 3059–3066. DOI: 10.1093/nar/gkf436 (cited on pages 6, 14, 31, 32, 37, 78, 79, 117, 138).

[Kawrykow et al. 2012] Kawrykow, A. et al. (2012). “Phylo: A Citizen Science Approach for Improving Multiple Sequence Alignment”. In: *PLOS ONE* 7.3, pp. 1–9. DOI: 10.1371/journal.pone.0031362 (cited on pages 41, 46, 47).

[Kececioglu and DeBlasio 2013] Kececioglu, J. and D. DeBlasio (2013). “Accuracy Estimation and Parameter Advising for Protein Multiple Sequence Alignment”. In: *Journal of Computational Biology* 20.4. ISSN: 1066-5277 (cited on pages 123–125, 131).

[Kemena and Notredame 2009] Kemena, C. and C. Notredame (2009). “Upcoming challenges for multiple sequence alignment methods in the high-throughput era”. In: *Bioinformatics* 25.19, pp. 2455–2465 (cited on page 32).

[Keul et al. 2017] Keul, F., M. Hess, M. Goesele, and K. Hamacher (2017). "PFASUM: a substitution matrix from Pfam structural alignments". In: *BMC Bioinformatics* 18.1, p. 293. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1703-z (cited on pages 4, 8, 76, 143).

[Khatib et al. 2011a] Khatib, F., S. Cooper, M. D. Tyka, K. Xu, I. Makedon, Z. Popović, D. Baker, and F. Players (2011a). "Algorithm discovery by protein folding game players". In: *Proceedings of the National Academy of Sciences* 108.47, pp. 18949–18953. DOI: 10.1073/pnas.1115898108 (cited on page 46).

[Khatib et al. 2011b] Khatib, F. et al. (2011b). "Crystal structure of a monomeric retroviral protease solved by protein folding game players". In: *Nature structural & molecular biology* 18.10, pp. 1175–1177 (cited on page 46).

[Kimura 1983] Kimura, M. (1983). *The neutral theory of molecular evolution*. Cambridge University Press (cited on page 36).

[Korpela et al. 2001] Korpela, E., D. Werthimer, D. Anderson, J. Cobb, and M. Lebofsky (2001). "SETI@home—Massively Distributed Computing for SETI". In: *Computing in Science & Engineering* 3.1, pp. 78–83. DOI: 10.1109/5992.895191 (cited on page 43).

[Krause and Vingron 1998] Krause, A and M Vingron (1998). "A set-theoretic approach to database searching and clustering". In: *Bioinformatics* 14.5, pp. 430–438. DOI: 10.1093/bioinformatics/14.5.430 (cited on page 77).

[Larkin et al. 2007] Larkin, M. et al. (2007). "Clustal W and Clustal X version 2.0". In: *Bioinformatics* 23.21 (cited on pages 35, 78, 126, 138).

[Larsson 2014] Larsson, A. (2014). "AliView: a fast and lightweight alignment viewer and editor for large datasets". In: *Bioinformatics* 30.22 (cited on pages 126, 127, 130, 133).

[Lassmann and Sonnhammer 2005] Lassmann, T. and E. L. L. Sonnhammer (2005). "Automatic assessment of alignment quality". In: *Nucleic Acids Research* 33.22 (cited on page 124).

[Lassmann et al. 2009] Lassmann, T., O. Frings, and E. L. L. Sonnhammer (2009). "Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features". In: *Nucleic Acids Research* 37.3 (cited on page 138).

[Lesk 2010] Lesk, A. (2010). *Introduction to protein science: architecture, function, and genomics*. 2nd edition. Oxford University Press (cited on pages 1, 45).

[Lesk 2013] Lesk, A. (2013). *Introduction to bioinformatics*. Oxford University Press (cited on pages 12, 13, 22, 25, 26, 55, 71, 75, 126).

[Lesk 2017] Lesk, A. (2017). *Introduction to genomics*. 3rd edition. Oxford University Press (cited on pages 1, 12).

[Lintott et al. 2008] Lintott, C. J. et al. (2008). "Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey". In: *Monthly Notices of the Royal Astronomical Society* 389.3, p. 1179. DOI: 10.1111/j.1365-2966.2008.13689.x (cited on page 43).

[Lipman et al. 1989] Lipman, D. J., S. F. Altschul, and J. D. Kececioglu (1989). “A tool for multiple sequence alignment”. In: *Proceedings of the National Academy of Sciences* 86.12, pp. 4412–4415 (cited on page 29).

[Liu et al. 2009] Liu, K., S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow (2009). “Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees”. In: *Science* 324.5934, pp. 1561–1564 (cited on page 33).

[Liu et al. 2010] Liu, Y., B. Schmidt, and D. L. Maskell (2010). “MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities”. In: *Bioinformatics* 26.16, pp. 1958–1964. DOI: 10.1093/bioinformatics/btq338 (cited on page 32).

[Löytynoja and Goldman 2010] Löytynoja, A. and N. Goldman (2010). “webPRANK: a phylogeny-aware multiple sequence aligner with interactive alignment browser”. In: *BMC Bioinformatics* 11.1 (cited on page 126).

[Löytynoja and Goldman 2005] Löytynoja, A. and N. Goldman (2005). “An algorithm for progressive multiple alignment of sequences with insertions”. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.30, pp. 10557–10562. DOI: 10.1073/pnas.0409137102 (cited on pages 15, 32, 33).

[Marx 2015] Marx, V. (2015). “Mapping proteins with spatial proteomics”. In: *Nature Methods* 12.9, pp. 815–819. ISSN: 1548-7091. DOI: 10.1038/nmeth.3555 (cited on page 44).

[McNaught and Wilkinson 1997] McNaught, A. D. and A. Wilkinson (1997). *IUPAC. Compendium of Chemical Terminology*. 2nd ed. (the "Gold Book"). Blackwell Scientific Publications, Oxford. DOI: 10.1351/goldbook (cited on page 1).

[Mevissen and Vingron 1996] Mevissen, H. T. and M. Vingron (1996). “Quantifying the local reliability of a sequence alignment”. In: *Protein Engineering* 9.2 (cited on page 124).

[Morgenstern 1999] Morgenstern, B (1999). “DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment.” In: *Bioinformatics* 15.3, pp. 211–218. DOI: 10.1093/bioinformatics/15.3.211 (cited on page 36).

[Mount 2004] Mount, D. W. (2004). *Bioinformatics: Sequence and Genome Analysis*. 2nd edition. Cold Spring Harbor (New York): Cold Spring Harbor Laboratory Press (cited on pages 1, 5, 14, 21, 22, 24, 29).

[Müller and Vingron 2000] Müller, T. and M. Vingron (2000). “Modeling amino acid replacement”. In: *Journal of Computational Biology* 7.6, pp. 761–776. DOI: 10.1089/10665270050514918 (cited on pages 4, 76, 77).

[Müller et al. 2002] Müller, T., R. Spang, and M. Vingron (2002). “Estimating Amino Acid Substitution Models: A Comparison of Dayhoff’s Estimator, the Resolvent Approach and a Maximum Likelihood Method”. In: *Molecular Biology and Evolution* 19.1, pp. 8–13. DOI: 10.1093/oxfordjournals.molbev.a003985 (cited on pages 4, 76, 77).

[Murzin et al. 1995] Murzin, A. G., S. E. Brenner, T. Hubbard, and C. Chothia (1995). "SCOP: a structural classification of proteins database for the investigation of sequences and structures". In: *Journal of Molecular Biology* 247.4, pp. 536–540 (cited on pages 79, 106).

[Nacke 2009] Nacke, L. (2009). "Affective ludology: Scientific measurement of user experience in interactive entertainment". In: (cited on pages 42, 50, 56, 64, 150).

[Needleman and Wunsch 1970] Needleman, S. B. and C. D. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." eng. In: *Journal of Molecular Biology* 48.3, pp. 443–453 (cited on pages 6, 14, 15).

[Nelesen et al. 2008] Nelesen, S. M., K. Liu, D Zhao, C. R. Linder, and T. J. Warnow (2008). "The Effect of the Guide Tree on Multiple Sequence Alignments and Subsequent Phylogenetic Analysis." In: *Pacific Symposium on Biocomputing*. Vol. 13. 2008, pp. 25–36 (cited on page 30).

[Ng et al. 2000] Ng, P. C., J. G. Henikoff, and S. Henikoff (2000). "PHAT: a transmembrane-specific substitution matrix". In: *Bioinformatics* 16.9, p. 760. DOI: 10.1093/bioinformatics/16.9.760 (cited on page 76).

[Notredame et al. 2000] Notredame, C., D. G. Higgins, and J. Heringa (2000). "T-Coffee: A novel method for fast and accurate multiple sequence alignment". In: *Journal of Molecular Biology* 302.1, pp. 205–217 (cited on pages 30–32, 35, 37).

[Ohsen and Zimmer 2001] Ohsen, N. von and R. Zimmer (2001). "Improving profile-profile alignments via log average scoring". In: *International Workshop on Algorithms in Bioinformatics*. Springer, pp. 11–26 (cited on page 36).

[Orbanes 2007] Orbanes, P. E. (2007). *Monopoly: The World's Most Famous Game–And How It Got That Way*. Da Capo Press (cited on page 41).

[Pearson 1991] Pearson, W. R. (1991). "Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms". In: *Genomics* 11.3, pp. 635–650. ISSN: 0888-7543. DOI: 10.1016/0888-7543(91)90071-L (cited on pages 5, 14, 21, 22, 67, 79, 80, 92, 93, 106, 107, 117).

[Pearson 1998] Pearson, W. R. (1998). "Empirical statistical estimates for sequence similarity searches". In: *Journal of Molecular Biology* 276.1, pp. 71–84 (cited on pages 21, 23).

[Pearson 2013] Pearson, W. R. (2013). "An introduction to sequence similarity ("homology") searching". In: *Current protocols in bioinformatics*, pp. 3–1 (cited on pages 5, 21, 23).

[Pearson and Lipman 1988] Pearson, W. R. and D. J. Lipman (1988). "Improved tools for biological sequence comparison". In: *Proceedings of the National Academy of Sciences* 85.8, pp. 2444–2448 (cited on pages 14, 21, 22, 35, 76, 117).

[Peplow 2016] Peplow, M. (2016). "Citizen science lures gamers into Sweden's Human Protein Atlas". In: *Nature Biotechnology* 34.5, pp. 452–453. ISSN: 1087-0156. DOI: 10.1038/nbt0516-452c (cited on page 45).

[Price et al. 2005] Price, G. A., G. E. Crooks, R. E. Green, and S. E. Brenner (2005). "Statistical evaluation of pairwise protein sequence comparison with the Bayesian bootstrap". In: *Bioinformatics* 21.20, pp. 3824–3831. DOI: 10.1093/bioinformatics/bti627 (cited on pages 6, 78–81, 83, 85, 92, 93, 101, 102, 107, 108, 123, 145).

[Project Discovery 2017] Project Discovery (2017). *Project Discovery*. Accessed 16.05.2017. URL: https://www.eveonline.com/discovery (cited on page 44).

[Raghava et al. 2003] Raghava, G., S. M. Searle, P. C. Audley, J. D. Barber, and G. J. Barton (2003). "OXBench: A benchmark for evaluation of protein multiple sequence alignment accuracy". In: *BMC Bioinformatics* 4.1, p. 47. ISSN: 1471-2105. DOI: 10.1186/1471-2105-4-47 (cited on pages 38, 39, 82, 117).

[Reese and Pearson 2002] Reese, J. and W. Pearson (2002). "Empirical determination of effective gap penalties for sequence comparison". In: *Bioinformatics* 18.11, pp. 1500–1507. DOI: 10.1093/bioinformatics/18.11.1500 (cited on pages 6, 14, 123).

[Rohl et al. 2004] Rohl, C. A., C. E. Strauss, K. M. Misura, and D. Baker (2004). "Protein structure prediction using Rosetta". In: *Methods in Enzymology* 383, pp. 66–93 (cited on page 45).

[Russell and Barton 1992] Russell, R. B. and G. J. Barton (1992). "Multiple protein sequence alignment from tertiary structure comparison: Assignment of global and residue confidence levels". In: *Proteins: Struct, Funct, Bioinf* 14.2, pp. 309–323. ISSN: 1097-0134. DOI: 10.1002/prot.340140216 (cited on pages 39, 118).

[Saitou and Nei 1987] Saitou, N. and M. Nei (1987). "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular Biology and Evolution* 4.4, pp. 406–425 (cited on pages 30, 34).

[Sauder et al. 2000] Sauder, J. M., J. W. Arthur, and R. L. Dunbrack Jr (2000). "Large-scale comparison of protein sequence alignment algorithms with structure alignments". In: *Proteins: Structure, Function, and Bioinformatics* 40.1, pp. 6–22 (cited on pages 39, 82).

[Sela et al. 2015] Sela, I., H. Ashkenazy, K. Katoh, and T. Pupko (2015). "GUIDANCE2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters". In: *Nucleic Acids Research* 43.W1 (cited on page 124).

[Shirts and Pande 2000] Shirts, M. and V. S. Pande (2000). "Screen Savers of the World Unite!" In: *Science* 290.5498, pp. 1903–1904. ISSN: 0036-8075. DOI: 10.1126/science.290.5498.1903 (cited on page 43).

[Siddiqui et al. 2001] Siddiqui, A. S., U. Dengler, and G. J. Barton (2001). "3Dee: a database of protein structural domains". In: *Bioinformatics* 17.2, pp. 200–201. DOI: 10.1093/bioinformatics/17.2.200 (cited on pages 39, 118).

[Sievers et al. 2011] Sievers, F. et al. (2011). "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega". In: *Molecular Systems Biology* 7.1. ISSN: 1744-4292 (cited on pages 30, 31, 37, 138).

[Simmons et al. 2017] Simmons, B. D. et al. (2017). “Galaxy Zoo: quantitative visual morphological classifications for 48000 galaxies from CANDELS”. In: *Monthly Notices of the Royal Astronomical Society* 464.4, p. 4420. DOI: 10.1093/mnras/stw2587 (cited on page 43).

[Smith and Waterman 1981] Smith, T. F. and M. S. Waterman (1981). “Identification of common molecular subsequences.” In: *Journal of Molecular Biology* 147.1, pp. 195–197 (cited on pages 6, 14, 19).

[Sneath and Sokal 1973] Sneath, P. and R. Sokal (1973). *Numerical Taxonomy*. San Francisco: W.H. Freeman and Company (cited on pages 30, 35–37).

[Söding 2004] Söding, J. (2004). “Protein homology detection by HMM–HMM comparison”. In: *Bioinformatics* 21.7, pp. 951–960 (cited on page 37).

[Song et al. 2015] Song, D., J. Chen, G. Chen, N. Li, J. Li, J. Fan, D. Bu, and S. C. Li (2015). “Parameterized BLOSUM Matrices for Protein Alignment”. In: *IEEE/ACM Trans Comput Biol Bioinform* 12.3, pp. 686–694. ISSN: 1545-5963. DOI: 10.1109/TCBB.2014.2366126 (cited on pages 76, 81, 92).

[Styczynski et al. 2008] Styczynski, M. P., K. L. Jensen, I. Rigoutsos, and G. Stephanopoulos (2008). “BLOSUM62 miscalculations improve search performance”. In: *Nature Biotechnology* 26.3, pp. 274–275. DOI: 10.1038/nbt0308-274 (cited on pages 4, 8, 76, 78–81, 83–85, 87, 91–93, 95, 101, 107).

[The Human Protein Atlas Blog 2016] The Human Protein Atlas Blog (2016). *The proteins are the building blocks of life*. Accessed 17.05.2017. URL: http://www.proteinatlas.org/blog/2016-03-08/project-discovery-a-contribution-to-science (cited on page 44).

[Thompson et al. 1999a] Thompson, J. D., F Plewniak, and O Poch (1999a). “BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs.” In: *Bioinformatics* 15.1, pp. 87–88. DOI: 10.1093/bioinformatics/15.1.87 (cited on pages 39, 125).

[Thompson et al. 1994] Thompson, J. D., D. G. Higgins, and T. J. Gibson (1994). “CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice”. In: *Nucleic Acids Research* 22.22 (cited on pages 13, 14, 33, 34, 36, 37).

[Thompson et al. 1999b] Thompson, J. D., F. Plewniak, and O. Poch (1999b). “A comprehensive comparison of multiple sequence alignment programs”. In: *Nucleic Acids Research* 27.13 (cited on pages 27, 125, 131).

[Thompson et al. 2005] Thompson, J. D., P. Koehl, R. Ripp, and O. Poch (2005). “BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark”. In: *Proteins: Struct, Funct, Bioinf* 61.1, pp. 127–136. ISSN: 1097-0134. DOI: 10.1002/prot.20527 (cited on pages 38, 39, 82, 117).

[Uhlén et al. 2005] Uhlén, M. et al. (2005). "A Human Protein Atlas for Normal and Cancer Tissues Based on Antibody Proteomics". In: *Molecular & Cellular Proteomics* 4.12, pp. 1920–1932. DOI: 10.1074/mcp.M500279-MCP200 (cited on page 44).

[Uhlen et al. 2010] Uhlen, M. et al. (2010). "Towards a knowledge-based Human Protein Atlas". In: *Nature Biotechnology* 28.12, pp. 1248–1250. ISSN: 1087-0156. DOI: 10.1038/nbt1210-1248 (cited on page 44).

[Uhlén et al. 2015] Uhlén, M. et al. (2015). "Tissue-based map of the human proteome". In: *Science* 347.6220. ISSN: 0036-8075. DOI: 10.1126/science.1260419 (cited on page 44).

[Van Walle et al. 2005] Van Walle, I., I. Lasters, and L. Wyns (2005). "SABmark–a benchmark for sequence alignment that covers the entire known fold space". In: *Bioinformatics* 21.7, pp. 1267–1268. DOI: 10.1093/bioinformatics/bth493 (cited on pages 38, 39, 82, 117).

[Vingron 1996] Vingron, M. (1996). "Near-optimal sequence alignment". In: *Current Opinion in Structural Biology* 6.3 (cited on page 124).

[Vingron and Argos 1990] Vingron, M. and P. Argos (1990). "Determination of reliable regions in protein sequence alignments". In: *Protein Engineering* 3.7 (cited on pages 124, 137).

[Wang et al. 2011] Wang, C., R.-X. Yan, X.-F. Wang, J.-N. Si, and Z. Zhang (2011). "Comparison of linear gap penalties and profile-based variable gap penalties in profile–profile alignments". In: *Computational Biology and Chemistry* 35.5, pp. 308 –318. ISSN: 1476-9271. DOI: 10.1016/j.compbiolchem.2011.07.006 (cited on page 14).

[Wang et al. 2012] Wang, C. K., U. Broder, S. K. Weeratunga, R. B. Gasser, A. Loukas, and A. Hofmann (2012). "SBAL: a practical tool to generate and edit structure-based amino acid sequence alignments". In: *Bioinformatics* 28.7 (cited on page 126).

[Wang and Jiang 1994] Wang, L. and T. Jiang (1994). "On the complexity of multiple sequence alignment". In: *Journal of Computational Biology* 1.4, pp. 337–348 (cited on pages 5, 6, 27, 29, 123).

[Waterhouse et al. 2009] Waterhouse, A. M., J. B. Procter, D. M. Martin, M. Clamp, and G. J. Barton (2009). "Jalview Version 2-a multiple sequence alignment editor and analysis workbench". In: *Bioinformatics* 25.9 (cited on page 126).

[Waterman et al. 1976] Waterman, M., T. Smith, and W. Beyer (1976). "Some biological sequence metrics". In: *Advances in Mathematics* 20.3, pp. 367 –387. ISSN: 0001-8708. DOI: 10.1016/0001-8708(76)90202-4 (cited on page 15).

[Wheeler and Kececioglu 2007] Wheeler, T. J. and J. D. Kececioglu (2007). "Multiple alignment by aligning alignments". In: *Bioinformatics* 23.13, pp. i559–i568 (cited on pages 30, 31).

[Willett et al. 2013] Willett, K. W. et al. (2013). “Galaxy Zoo 2: detailed morphological classifications for 304122 galaxies from the Sloan Digital Sky Survey”. In: *Monthly Notices of the Royal Astronomical Society* 435.4, p. 2835. DOI: 10.1093/mnras/stt1458 (cited on page 43).

[Willett et al. 2017] Willett, K. W. et al. (2017). “Galaxy Zoo: morphological classifications for 120000 galaxies in HST legacy imaging”. In: *Monthly Notices of the Royal Astronomical Society* 464.4, p. 4176. DOI: 10.1093/mnras/stw2568 (cited on page 43).

[Wright 2015] Wright, E. S. (2015). “DECIPHER: harnessing local sequence context to improve protein multiple sequence alignment”. In: *BMC Bioinformatics* 16.1, p. 322 (cited on pages 6, 37).

[Young and Healy 2003] Young, N. D. and J. Healy (2003). “GapCoder automates the use of indel characters in phylogenetic analysis”. In: *BMC Bioinformatics* 4.1, p. 6. ISSN: 1471-2105. DOI: 10.1186/1471-2105-4-6 (cited on page 14).

[Zhan et al. 2015] Zhan, Q., Y. Ye, T.-W. Lam, S.-M. Yiu, Y. Wang, and H.-F. Ting (2015). “Improving multiple sequence alignment by using better guide trees”. In: *BMC Bioinformatics* 16.5, S4. ISSN: 1471-2105. DOI: 10.1186/1471-2105-16-S5-S4 (cited on page 30).

[Zhang et al. 1998] Zhang, Z., W. Miller, A. A. Schäffer, T. L. Madden, D. J. Lipman, E. V. Koonin, and S. F. Altschul (1998). “Protein sequence similarity searches using patterns as seeds”. In: *Nucleic Acids Research* 26.17, pp. 3986–3990. DOI: 10.1093/nar/26.17.3986 (cited on page 23).