Fast methods for metagenomic sequence search & annotation

Dissertation

for the award of the degree
"Doctor rerum naturalium"
of the Georg-August-Universität Göttingen

within the doctoral program in Computer Science
of the Georg-August University School of Science (GAUSS-PCS)

submitted by
Milot Mirdita
Göttingen, 2021

Thesis Advisory Committee

- Dr. Johannes Söding

  Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry

- Prof. Dr. Stephan Waack

  Theoretical Computer Science and Algorithmic Methods, Institute of Computer Science,
  Georg-August University Göttingen

Members of the Examination Board

1st Reviewer: Dr. Johannes Söding

Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry

2nd Reviewer: Prof. Dr. Stephan Waack

Theoretical Computer Science and Algorithmic Methods, Institute of Computer Science,
Georg-August University Göttingen

Further members of the Examination Board

- Prof. Dr. Jan de Vries

  Department of Applied Bioinformatics, Institute for Microbiology and Genetics,
  Georg-August University Göttingen

- Prof. Dr. Michael Altenbuchinger

  Department of Medical Bioinformatics, University Medical Center Göttingen

- Prof. Dr. Burkhard Morgenstern

  Department Bioinformatics, Institute for Microbiology and Genetics,
  Georg-August University Göttingen

- Dr. Juliane Liepe

  Quantitative and Systems Biology, Max Planck Institute for Biophysical Chemistry

Date of oral examination: February 21st , 2022

Acknowledgments

I am deeply grateful to my mentor and supervisor Dr. Johannes Söding for your support over the years. Our discussions have very deeply shaped my views of science, society and ethics.

I want to deeply thank Prof. Dr Stephan Waack for accompanying my doctoral research as part of my thesis advisory committee and for agreeing to review this work,

Furthermore, I am thankful to Dr. Juliane Liepe, Prof. Dr. Jan de Vries, Prof. Dr. Michael Altenbuchinger, Prof. Dr. Burkhard Morgenstern for agreeing to take part in the thesis examination.

I would like to thank the many administrative staff and colleagues at the MPI-BPC, University of Göttingen and GWDG for their help and assistance.

I am eternally grateful towards my best friend Prof. Dr. Martin Steinegger. In the same breath, I want to also thank Dr. Eli Levy Karin, Dr. Christian Roth and Dr. Nikolaos Papadopoulos. I am deeply grateful that I was lucky enough to work together on many projects with the former two, and deeply regretful that I have not yet had a chance to work on the same project with the latter two. In addition to your friendships, I also want to thank you for the reviews that immeasurably improved this work.

The members of the Södinglab will forever have a special place in my heart. I want to especially thank Ruoshi Zhang, Dr. Salma Sohrabi-Jahromi and Dr. Christian Roth for the many evenings we shared, cuisines we explored and insightful discussions we had. And a thanks to all the participants of the board game evenings for brightening my time in Göttingen – first among them Dr. Saikat Banerjee.

I am indebted to the teachers who awakened my interest in science and put me onto a path towards bioinformatics.

I want to thank my parents, Have and Dede Mirdita, for always instilling the value of education in me.

And lastly,
I want to thank my wife Mỹ Linh Huỳnh
for believing in me and sharing our life together.

Summary

The past two decades have seen the development of metagenomics, the study of genes and genomes of multiple organisms simultaneously. In contrast to traditional genomic techniques, which require isolating and growing individual organisms in the lab, in metagenomics, samples are directly taken from the environment, sequenced and then analyzed *in silico*. Modern sequencing techniques have enabled high throughput read-out of DNA and RNA of microorganism communities in marine, soil, gut and many other environments.

The plethora of data generated using these techniques poses a major challenge for existing computational techniques. This burden translates directly to computational run times and the cost of resources required to carry out metagenomic analyses. Thus, computational methods developed for metagenomic analysis require exceptional efficiency and speed. At the same time, metagenomic studies become relevant for more and more fields of research, requiring that techniques be suited for a wide range of scientific disciplines.

In this work, I present three methods I developed to address the throughput bottlenecks of data analysis in metagenomics. (1) The MMseqs2 webserver is a user-friendly extension of the popular homology search method MMseqs2 designed for non-expert bioinformaticians. I accelerated MMseqs2 to process single queries much more quickly and introduced an API to enable MMseqs2's use in web applications. (2) MMseqs2 taxonomy is a method for fast and accurate taxonomy assignment of metagenomic contigs. (3) ColabFold is a method to make the groundbreaking AlphaFold2 protein structure predictions widely accessible, accelerating its input sequence alignment generation and improving its accuracy by assembling a novel database enriched with metagenomic sequences from a multitude of datasets.

These methods improve upon the state-of-the-art by introducing novel algorithms and accelerating previous ones – such that previously infeasible analyses become possible – and making our metagenomic toolbox accessible to users of a wide range of skill levels.

Contents

List of commonly used abbreviations

BLOSUM Blocks Substitution Matrix. 9

CASP Critical Assessment of Structure Prediction. 17, 18

CRISPR Clustered Regularly Interspaced Short Palindromic Repeats. 4

DNA deoxyribonucleic acid. 5, 6, 13, 16, 125

GTDB Genome Taxonomy Database. 16, 123

HMM Hidden Markov Model. 10

ICVT International Committee on Taxonomy of Viruses. 15

MSA Multiple Sequence Alignment. 1, 3, 9, 10, 18, 118, 121, 122, 124

NCBI National Center for Biotechnology Information. 15

NHGRI National Human Genome Research Institute. 13

NW Needleman-Wunsch. 8, 9

PAM Point Accepted Mutation. 9

PDB Protein Data Bank. 6

PSSM Position Specific Scoring Matrix. 10, 18

RNA ribonucleic acid. 5, 16

SIMD Single Instruction Multiple Data. 12, 13

SRA Sequence Read Archive. 1, 3, 14, 123

SW Smith-Waterman. 8, 9

1 Introduction

In the last decades, computational methods have revolutionized the biological sciences. They have become indispensable to deal with the avalanche of data produced by state-of-the-art biological experiments.

The sequence similarity search method BLAST [Altschul et al., 1990] is among the computational methods that arguably have had the biggest impact on biological sciences to date. BLAST and PSI-BLAST [Altschul et al., 1997] have been cited close to two hundred thousand times. Homology search methods find sequences similar to a given query sequence in a larger sequence collection. Similarity is defined in a biological sense; two sequences are similar if they share a sufficient number of identical or similar characters. Sufficient levels of sequence similarity are evidence for a common evolutionary origin, and such sequences are named homologous. In section 1.2 I will describe algorithms for homology search in detail.

In the last decade several methods have been developed that far eclipse BLAST in speed and sensitivity – the ability to detect remotely homologous sequences with low similarity to the query. At the forefront of this new wave of homology search methods lie DIAMOND [Buchfink et al., 2015] and MMseqs2 [Steinegger and Söding, 2017]. I want to highlight two recent groundbreaking projects that were enabled by DIAMOND and MMseqs2:

A notable example of DIAMOND's ability to process metagenomic data was the recent, extremely parallelized search where the whole Sequence Read Archive (SRA) [Sayers et al., 2021a] was systematically queried for markers of viral proteins [Edgar et al., 2020]. The SRA stores most published genomic experiments and consists of multiple petabytes of nucleotide sequences (see section 1.4). It was distributed to a large set of cloud computing resources and the authors could achieve extremely low costs per processed set in the SRA.

Recently, AlphaFold2 [Jumper et al., 2021a] was able to predict protein structures from their sequences with accuracies comparable to experimental structure determination methods. To compute high-quality protein structures, AlphaFold2 requires diverse Multiple Sequence Alignments (MSAs) as input. To build these MSAs enormous protein reference catalogs [Mirdita et al., 2017, Mitchell et al., 2020, Jumper et al., 2021a] are queried for similar sequences by sensitive search methods. MMseqs2's clustering capabilities have enabled building these enormous protein reference catalogs, resulting in protein databases of sizes in the order of multiple billion protein sequences. In section 1.6 I describe the challenges of computational protein structure prediction.

Scope of this work

The main goal of my doctoral research was to develop efficient computational methods for the analysis of large amounts of biological data. Specifically, I wanted to make these available as high-quality software for users with a wide range of skill levels in bioinformatics. Additionally, I have been involved in the development of MMseqs2 and other methods within its ecosystem since 2014. Since then, I have contributed to extending MMseqs2 and leveraged its capabilities. As MMseqs2 is a foundational method for this work, I will present it in detail in section 1.3.

Main publications

I want to highlight three projects in this work, to which I contributed in a leading role:

MMseqs2 App and Server. The original MMseqs2 toolkit was developed for expert users who are savvy in running software through a shell and process its output by writing shell scripts. It has therefore been my goal to make MMseqs2 available for scientists not comfortable with using a command line. To this end, I developed a version of MMseqs2 that can be run as either a traditional desktop application or as an easily deployable webserver.

As part of this work, I accelerated MMseqs2 searches for single- or small sets of queries, by reducing the overhead between module invocations (see 1.3 for an introduction to MMseqs2's architecture). Such searches are more common for users who want to interactively investigate individual proteins. In addition, I greatly expanded the facilities for precomputing data structures for searches, so that these can directly be read from disk. These changes allow computing search results within milliseconds to seconds. In this manuscript, I also show that MMseqs2 can be used for highly efficient searches against profile databases (e.g., PFAM [Mistry et al., 2021]).

I further expanded the MMseqs2 webserver API to serve two additional use-cases. In collaboration with the PredictProtein authors (Bernhofer et al. [2021], see section 6.2 for the abstract), we expanded the MMseqs2 webserver to quickly return multiple sequence alignments for the various downstream protein property prediction methods that are available in PredictProtein. Up to now, the MMseqs2 server has processed over a hundred thousand MSAs for PredictProtein users. Similarly, I extended the MMseqs2 server to serve diverse MSAs for use in protein structure prediction in ColabFold (see below).

Chapter 2 includes the manuscript for the MMseqs2 desktop and webserver app.

MMseqs2 Taxonomy. Annotation of unknown sequences is a common task in bioinformatics. In metagenomics, communities of diverse microorganisms are sampled directly from their native environments. Determining the taxonomic identity of each sequence from these samples allows many useful downstream analyses. For example, in clinical use a taxonomic analysis of a metagenomic sample from a patient could reveal the disease-causing agent. Existing taxonomy assignment methods are well suited to assign taxonomic identity to the short (often 2×150 base pair long) reads produced in paired-end sequencing experiments. Reads from these experiments are often assembled into longer contigs.

In this manuscript, we extended MMseqs2 to exploit the additional information available in contigs in order to assign reliable taxonomic labels. We compare MMseqs2 taxonomy with the state-of-the-art method CAT [Von Meijenfeldt et al., 2019] and show that we can assign taxonomic identity much faster at comparable quality. MMseqs2 taxonomy is now being used in the machine learning based binning method SemiBin [Pan et al., 2021]. See section 1.5 for a more in-depth introduction into taxonomy and chapter 3 for the manuscript.

ColabFold. The public release of AlphaFold2 [Jumper et al., 2021a] has started a new era in structural biology. For the first time, an *in silico* protein structure prediction method was shown to predict structures from protein sequences at an accuracy nearly indistinguishable from experimental structure determination methods. However, its heavy computational requirements raised concerns about its usability for many biologists. We leveraged the MMseqs2 Server's MSA generation capability to build ColabFold. ColabFold [Mirdita et al., 2021] replaces AlphaFold2's MSA generation stage to generate highly diverse MSAs in a matter of seconds to minutes instead of multiple hours. With the help of the free-to-access GPUs provided by Google Colaboratory, we could provide access to AlphaFold2's predictions to a wide community of researchers. Additionally, we pioneered modelling protein complexes with AlphaFold2. To-date our methods have been used to process many hundreds of thousands MSAs. See chapter 4 for the manuscript.

Additional publications

In addition to the main projects of my doctoral research, I have been involved in several other projects. The following is a brief presentation of them.

HH-suite3. In 2019 we released a new version of the HH-suite [Steinegger et al., 2019a], one of the most sensitive homology search methods to date. HH-suite leverages the Uniclust databases [Mirdita et al., 2017] for deep annotations across vast evolutionary time frames. Continuing development of HH-suite's databases was only possible due to the fast clustering capabilities of MMseqs2. HH-suite is also available within the MPI Bioinformatics Toolkit (Gabler et al. [2020], see section 6.1 for the abstract), an easy-to-use web server for sequence analysis. See section 5.2 for the manuscript.

Plass. With the rapid growth of the SRA, we observed that state-of-the-art assembly tools produced fragmented and low-quality assemblies on highly diverse metagenomic datasets, limiting the amount of sequences that can be gleaned from such datasets. Specifically, the majority of these tools rely on exact-kmer matching (De-Bruijn Graph Assemblies), which deteriorate significantly as the number of species increase and the sequence coverage drops. With Plass [Steinegger et al., 2019b], we showed that we could assemble translated protein fragments through the alternative approach of overlap-assembly and thus could extract many times the number of proteins from metagenomic and metatranscriptomic datasets than state-of-the-art methods. See section 5.1 for the manuscript.

MetaEuk. We observed that even though the availability of datasets enriched for eukaryotic life-forms is increasing rapidly [Carradec et al., 2018], studying their genomes remained difficult due to the complex intron-exon structures of eukaryotic genes. We developed MetaEuk, a reference-based gene finder and annotator, which takes into account the unique features of eukaryotic genes by efficiently examining all possible exon sets [Levy Karin et al., 2020]. MetaEuk has been integrated as the default gene predictor in the highly popular BUSCO tool suite for assessing assembly completeness [Manni et al., 2021]. See section 5.3 for the manuscript.

SpacePHARER. CRISPR spacers (Clustered Regularly Interspaced Short Palindromic Repeats) are short viral genome fragments integrated into a majority of bacterial and archaeal genomes that guide their adaptive immune response [Hille et al., 2018]. With SpacePHARER [Zhang et al., 2021], we developed a method to exploit these sequences to identify host-phage relationships reliably and faster than previously possible. Here, MMseqs2 was optimized for searching short, translated protein fragments. See section 5.4 for the manuscript.

1.1 The central dogma of biology

The Central Dogma of Biology [Cobb, 2017] describes the process of information flow at the heart of biology. This process starts with deoxyribonucleic acid (DNA), the blueprint of life that is passed from generation to generation. DNA is a long polymer made up of four different bases, abbreviated into one of four letters: A for adenine, T for thymine, G for guanine and C for cytosine. To produce proteins, DNA is first transcribed by polymerase proteins into single-stranded strands of nucleic acids called ribonucleic acid (RNA).

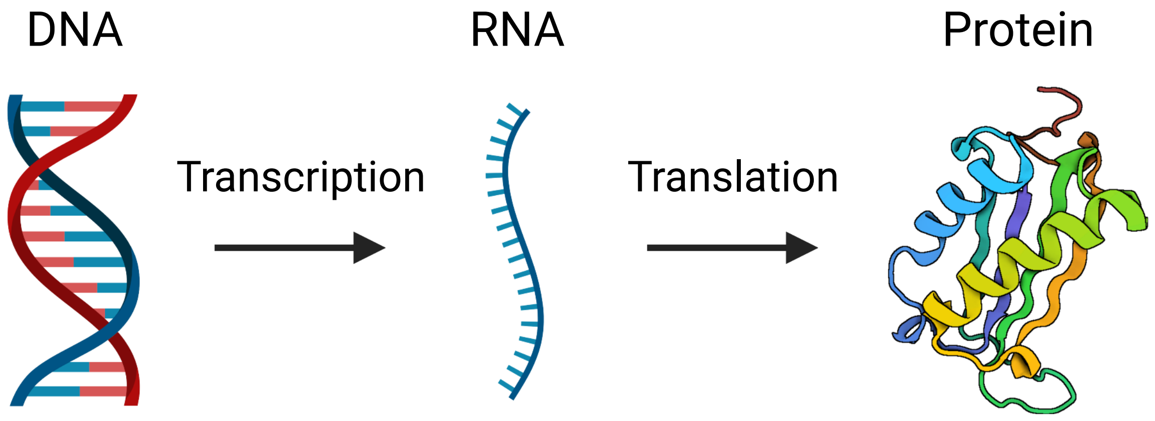


Figure 1.1: The central dogma describes a process of information flow from DNA (left) getting transcribed to RNA (middle) and finally translated to a protein (right). Example protein structure from PDB 1DPT [Sugimoto et al., 1998, 1999]. Created with BioRender.com.

Like DNA, RNA consists of four nucleic acids. Unlike DNA, RNA uses the close chemical relative Uracil (with the letter U) instead of thymine. The process of generating RNA from a DNA template is called transcription and serves to spatially and temporarily split "blueprint" read-out from its use to manufacture proteins.

Ribosomes attach to the RNA strand and translate the RNA to a chain of amino acids. Ribosomes read the RNA in groups of three nucleotides (called a codon) and place based on the 3-codon letters the corresponding amino acid into the new chain to form a protein.

DNA & amino-acid sequences

DNA and Protein sequence determination began with the invention of degradation methods where one nucleic- or amino acid could be removed and read at a time [Edman et al., 1950, Sanger and Coulson, 1975]. Soon after Edman-degradation was established, hundreds of thousands of residues of proteins were known. The first protein databases were compiled into books and published for the community by scientists such as Margret Dayhoff [Hersh et al., 1967, Strasser, 2010].

As sequencing methods became more efficient, it did not take long for these to become too large to publish in print and were only feasible as electronic files. The GenBank was established in 1982 [Burks et al., 1985] to collect all DNA sequence data produced from sequencing experiments. For protein sequences, the Swiss-Prot database was established in 1986 with the goal of making all known protein sequences digitally available [Bairoch and Boeckmann, 1991]. In its current release, GenBank stores trillions of base pairs [Sayers et al., 2021b] and the UniProtKB/Swiss-Prot+TrEMBL contains > 190 mil. protein sequences [Bateman et al., 2021].

The rapid growth of sequence databases is fueled by next-generation sequencing techniques that enable high-throughput read out of DNA sequences (as reviewed in e.g., Hu et al. [2021]). While currently infeasible, direct high throughput read-out of protein sequences might soon become possible with e.g., Nanopore based protein sequencing [Brinkerhoff et al., 2021].

Protein structures

Proteins have been described as one of the basic building blocks of life [Marth, 2008]. They spontaneously fold into three-dimensional conformations. Such structures include catalytic sites, binding sites, exposed and folded areas and more. Thus, revealing the structures proteins take under different biological conditions is crucial for understanding their function.

Fig. 1.1 (right) shows a ribbon-diagram [Richardson, 2000] of the Protein Data Bank (PDB) deposited protein 1DPT – a human d-dopachrome tautomerase protein. Ribbon-diagrams highlight secondary structure elements, in the foreground two α-helices, while the background contains four parallel β-sheets. This protein also oligomerizes into a hexamer, which is shown as an example for a protein complex in figure 1.2. Here, the protein shown in Fig. 1.1 (right) is repeated six times to form a homo-hexamer.

This example shows the various levels at which protein structures are organized: (1) The protein's primary structure is the sequence of its constituting amino acids. (2) Its secondary structure is typically defined by three local structure element types: α-helices, β-sheets and loops (finer grained classifications such as the eight DSSP states also exist [Kabsch and Sander, 1983]). (3) The tertiary structure of a protein defines the coordinates in three dimensions. (4) The quaternary structure is defined by the three-dimensional arrangement of multiple protein chains that form a protein complex.

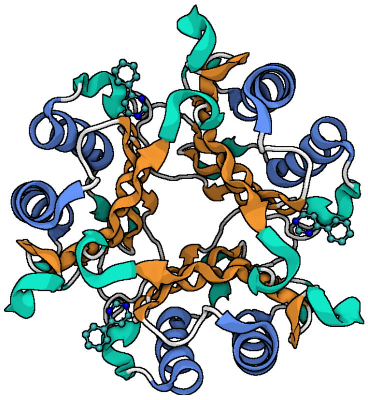


Figure 1.2: PDB Entry 3KAN [Zierow and Lolis, 2009, Rajasekaran et al., 2014]. Homo-hexamer of the tautomerase shown previously. Created with BioRender.com.

Since the early days of molecular biology, much effort has been invested in solving protein structures, which can be seen in the over 184 000 structures that are now deposited in the PDB. The PDB [Berman et al., 2003] is a union of international organizations to ensure that solved protein structures are openly accessible to all scientists.

1.2 Homology searches

In biology, identifying similarities and differences between sequences of genes and proteins is a powerful way to study their function. If for example, the same protein sequence is found among all organisms in a specific environment, however, is absent in closely related organisms who do not share that environment, one can postulate this protein may be important for survival under certain ecological conditions. Similarly, a part of a protein coding gene that is highly conserved even among sequences of evolutionarily distant species could be the active site of that protein. Lastly, if one has a set of carefully annotated sequences, laboriously established annotations about the members can be transferred to novel sequences, if these show sufficient similarity to some of the members of that collection (also known as homology-based inference of annotations).

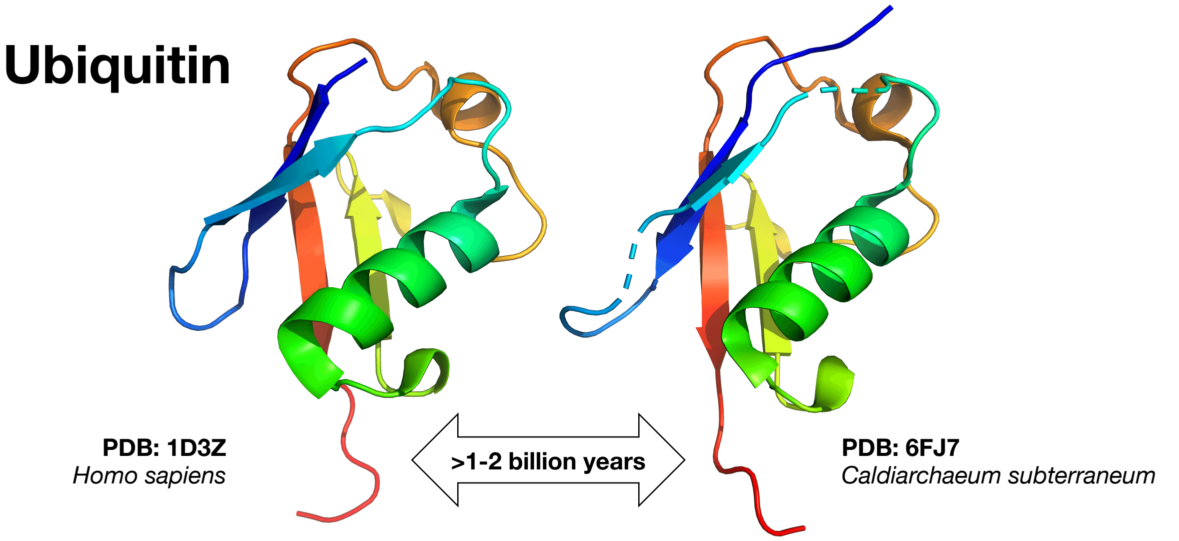
It is thus of great value to identify homology between sequences through sequence similarity. However, it is worth noting that the term "sequence homology" refers to sequences that share a joint ancestor, while "sequence similarity" refers to closeness under some (biochemical) metric. Thus, two sequences can be homologous but have some (even many) residues with very low sequence similarity. Moreover, it has been shown that some protein sequences can share similarities even though they are not homologous [Krishna and Grishin, 2004]. Finally, homology can be further categorized based on the evolutionary scenario (e.g., gene duplications, speciations, etc.).

A string is a sequence of characters from a finite alphabet and understandable by computers. This definition of strings fits both DNA/RNA and protein sequences as they have alphabets of sizes 4 and 20 respectively. Various algorithms were developed to compare and search these strings. In the following, I will briefly discuss important algorithms for aligning biological sequences.

Sequence alignments unravel the evolutionary history, as sequences that are similar are assumed to be evolutionary conserved and share common ancestral sequences. Fig 1.3 shows conserved proteins from *Homo sapiens* and the archaeon *Caldiarchaeum subterraneum*. As the last eukaryotic common ancestor is estimated to have existed over 1-2 billion years ago [Eme et al., 2014], the common ancestor of these two structures must have existed some additional time before that. The pairwise sequence alignment shown in the bottom of the figure has a sequence identity of 32.8%, showing that alignments can detect both evolutionary and functional conservation across billions of years.

Pairwise sequence alignments

For the first protein sequences, pairwise alignments were still possible through visual inspection, as only a few highly conserved homologs of a small number of proteins were known (e.g., the 10 homologous Cytochrome c proteins in Hersh et al. [1967]). However, it was clear that efficient computational solutions would be soon needed, as a naïve solution for optimal pair-



```
1D3Z         10 GKTITLEVEPSDTIENVKAKIQDKEGIPPDQQRLIFAGKQLEDGRTLSDYNIQKESTLHLVLRLRGG      76
                |..:.|||.|:.|:..|:.|:...:.:|||..||.:.|:.|:|..||....:.......|:.|..||
6FJ7         14 GSPLELEVAPNATVGAVRTKVCAMKKLPPDTTRLTYKGRALKDTETLESLGVADGDKFVLITRTVGG      80
```

Figure 1.3: Comparison of the human ubiquitin protein [Cornilescu et al., 1999, 1998] and the homologous one from an archaeon [Wojtynek et al., 2018, Fuchs et al., 2018]. Created with PyMol Open-Source (`pymol.org`), EMBOSS-Water [Madeira et al., 2019] and Sequence Manipulation Suite [Stothard, 2000].

ing of two sequences would have an exceedingly large search space. Algorithms based on dynamic programming were introduced to solve the pairwise alignment problem efficiently and elegantly [Eddy, 2004]. In dynamic programming, problems are broken down into (mathematically) optimally solvable independent sub-problems. These sub-problems are usually described with a recurrence equation. The same sub-problems can reoccur; thus, their solutions are memorized (e.g., tabulated in a grid). The final optimal solution is assembled from the solutions of the sub-problems.

One of the earliest (if not the first) applications of dynamic programming to pairwise comparison of sequences was the Needleman-Wunsch (NW) algorithm [Needleman and Wunsch, 1970]. NW and the closely related Smith-Waterman (SW) algorithm [Smith and Waterman, 1981] are still among the most important and widely used pairwise alignment algorithms. The former produces optimal global alignments, where all residues of each of the two sequences are included in the alignment. The latter produces optimal local alignments which do not necessarily include all residues, but rather only the most conserved parts of each sequence.

The recurrence equation for the NW and SW algorithms is shown in equation 1.1. M is a function returning the substitution or match/mismatch score between character i of sequence Q and character j of sequence T. In the simplest case, the algorithm gives a score of 1 for an exact match and a score of -1 for a mismatch. This is also called identity scoring. The function g returns a gap score. In the simplest case -1 could be used. The solutions to previous calls of the function $S(i,j)$ are memorized to avoid solving the sub-problems $S(i-1,j-1)$, $S(i-1,j)$ and $S(i,j-1)$ repeatedly.

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + M(Q_i,T_j), & \text{(match/mismatch)} \\ S(i-1,j) + g(Q_i,\text{–}), & \text{(deletion)} \\ S(i,j-1) + g(\text{–},T_j), & \text{(insertion)} \\ 0 & \text{(only in Smith-Waterman).} \end{cases} \tag{1.1}$$

To compute the alignment score of a query sequence Q and target sequence T, the two sequences are arranged on the two sides of a grid (usually with an additional first row and first column). The first row and column are initialized with zeros in the SW case, to allow an alignment to begin anywhere in either sequence. For NW the first row and column are initialized with increasing gap costs. The dynamic programming recurrence equation is computed for every cell. In SW the cell with the highest score is stored and marks the end of the alignment. In NW the cell in the bottom-right corner marks the end of the alignment.

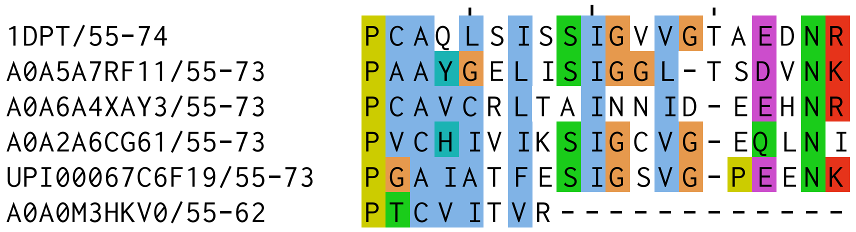
During each cell update the direction that was taken (left for deletion, top-left for match/mismatch, right for insertion) to compute the maximum is stored in a traceback matrix. From the end cell of the alignment the path is then traced back. If the top-left direction is chosen, the two corresponding letters in Q and T are aligned with each other. If either the left or top path is chosen a gap character (usually '–') is introduced and aligned with the corresponding sequence letter. In SW the alignment stops once the first zero is encountered. In NW the alignment continues to the top-left cell.

Affine gap costs. The algorithm as described scores many one-character gaps, just as highly as – biologically much more likely – fewer, but longer gaps. Affine gap costs were introduced to penalize the former. Affine gap costs separate the costs for opening a gap, so a higher cost can be "paid" a single time, and the costs for extending gaps, where a lower cost can be repeatedly paid. Gotoh [1982] introduced an efficient formulation of this algorithm.

Substitution matrices. Using identity scoring (e.g., 1 for matches and -1 for mismatches) is usually insufficient for amino acids, as for biological sequences some amino acids exchanges are more likely than others. This might be due to chemical relatedness, hydrophobicity, acidity and other properties. Amino acid substitution matrices were created that assign each pair of amino acids a score for the likelihood of substitution in evolutionarily related sequence families. One of the first such matrices was the Point Accepted Mutation (PAM) matrix [Dayhoff et al., 1978]. Later the Blocks Substitution Matrix (BLOSUM) family of matrices was introduced [Henikoff and Henikoff, 1992]. To create the BLOSUM matrices, amino acid substitution frequencies in highly conserved protein families were counted and turned into log-odd scores.

Multiple sequence alignments

Homologous sequences from multiple sources can also be aligned as MSAs. Here, each individual sequence is placed into a separate row and evolutionary conserved residues from each sequence are aligned into columns. Gap characters indicate insertions or deletion. As producing optimal MSAs was found to be *NP*-hard (e.g., in Elias [2006]), MSA generation methods use heuristics to produce MSAs in reasonable time. Figure 1.4 shows a cropped region of the MSA of the sequence of the tautomerase shown in Figure 1.1 and multiple related sequences from the UniProt [Bateman et al., 2021].



```
1DPT/55-74            PCAQLSISSIGVVGTAEDNR
A0A5A7RF11/55-73      PAAYGELISIGGL-TSDVNK
A0A6A4XAY3/55-73      PCAVCRLTAINNID-EEHNR
A0A2A6CG61/55-73      PVCHIVIKSIGCVG-EQLNI
UPI00067C6F19/55-73   PGAIATFESIGSVG-PEENK
A0A0M3HKV0/55-62      PTCVITVR------------
```

Figure 1.4: Cropped MSA of 1DPT against many UniRef sequences. Created with Jalview [Waterhouse et al., 2009].

An MSA can be the starting point for many different analyses, such as phylogenetic analysis [Feng and Doolittle, 1987] or determination of contacts within a protein [Göbel et al., 1994]. Protein families are usually provided as MSAs (e.g., PFAM [Mistry et al., 2021]). MSAs of sufficient quality continue to be important in AlphaFold2 [Jumper et al., 2021a] to produce highly accurate predicted protein structures, although around 30 sufficiently diverse sequences are often enough.

Sequence profiles. From multiple sequence alignments various forms of profiles can be computed that can be used for much more sensitive pairwise sequence searches. A simple form of sequence profile is the Position Specific Scoring Matrix (PSSM), where the $M \times N$ large MSA (M being the number of sequences and N the length of the MSA) is reduced to alphabet size $\times$ N. This allows during a pairwise alignment to create a position specific substitution matrix for each MSA column. Figure 1.5 shows a sequence logo [Schneider and Stephens, 1990], where the height of each letter indicates the likelihood of it to appear in this position. Highly conserved columns contain only a single tall letter.

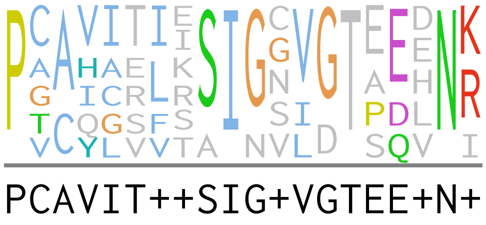


Figure 1.5: Sequence logo of 1DPT MSA from Fig. 1.4. Created with Jalview.

Hidden Markov Model (HMM) based profiles furthermore add various transition scores to the sequence profiles. This allows to explicitly model transition probabilities from match-to-match state, match-to-deletion and so on. Software like HMMER [Eddy, 2011] and HHblits [Steinegger et al., 2019a] use profile-HMMs for highly sensitive homology searches.

1.3 Current methods for homology search

Due to their quadratic runtime complexity, Needleman-Wunsch and Smith-Waterman can only be applied to small collections of sequences. However, as reference databases began undergoing rapid growth in the 1980s, faster heuristics were required to keep up. FASTA [Lipman and Pearson, 1985] and BLAST [Altschul et al., 1990] were some of the early methods that could deal with the growing databases. For reference, the Intel 386 and 486 CPUs were state-of-the-art in 1985 and 1990 respectively. They had 12MHz to 20MHz clock speeds and RAM amounts between 1MB to 16MB.

Today, reference databases contain hundreds of millions of sequences, while protein catalogs mined from metagenomics contain tens of billions of sequences. Workstations can contain hundreds of CPU cores and terabytes of RAM. Modern hardware features have enabled the development of much faster algorithms to handle the enormous amount of steadily growing data.

In the following I will focus on two methods that introduced novel algorithms and were able to exploit modern hardware to face the challenges of metagenomic data analyses.

MMseqs2

MMseqs2 [Steinegger and Söding, 2017] is one such method that profits from advances in hardware. Specifically, MMseqs2 uses the large amount of RAM available on modern systems to build efficient index structures of the reference databases. These allow fast and efficient queries to identify likely homologous candidates and discard likely unrelated ones. As MMseqs2 is used in every project in this work, I will provide an introduction into its software architecture and core algorithms.

At its core MMseqs2 facilitates batch processing through exchange of its data exchange formats (“databases”, see below), between modules implementing different algorithms. These consume specific input databases and emit new output databases containing the results of the respective algorithms.

For example, during a homology search a sequence file containing proteins in FASTA format is converted to an amino acid database. This database is given together with a similar database of reference sequence to the prefiltering module (`prefilter`) to quickly identify likely homologous pairs. The prefilter’s output is a new prefiltering database containing only these pairs. The two sequence databases and the prefiltering database are then used in the alignment module (`align`). It aligns the prefiltered pairs with the sensitive but much slower Smith-Waterman-Gotoh algorithm, identifies homologous pairs and emits them in a new alignment database. In the last step, the `convertalis` module converts the alignment database to a user defined human-readable output format.

This modularized architecture enables reuse of existing modules for new applications. Thus, only modules specific to a new application have to be implemented.

MMseqs2 databases. The MMseqs2 database format was built to avoid many pitfalls with processing a large number of small files. A typical sequence database contains tens to hundreds of millions of sequence entries. Processing these as separate files could lead to various issues, such as file-system slowdown and system-call overheads.

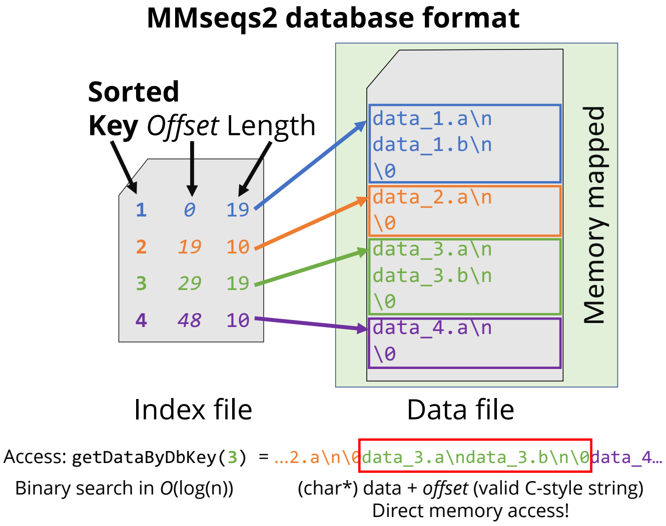


Figure 1.6: Details of the MMseqs2 database format. The database format was introduced for fast, direct access to many small entries.

Instead, we store all these entries into a single file with each entry separated from the next by a null byte character. Additionally, we store for each entry an identifier (key), its position (byte-offset) in the file and the entry length in a separate file (index file). The lines of the index file are sorted by keys. Thus, entries can be quickly found through a binary search (see Figure 1.6 for a schematic description). When a database is used, it is typically read using *memory mapping*, a mechanism provided by the operating system which allows reading the file as if it was a string in RAM.

Prefilter. The prefiltering module implements one of the most important algorithms in MMseqs2. The prefilter's main objective is to quickly identify two consecutive k-mer hits on the same diagonal above a given score threshold. k-mers are k characters-long sub-strings of a sequence. MMseqs2 precomputes a data structure to store every occurrence of a k-mer in a target database together with the position within its sequence. With this data structure, k-mer hits on the same diagonal ($i - j$, where i and j are the start positions of the respective k-mer in the query and target sequence) can be efficiently identified. Demanding double-consecutive hits increases selectivity by rejecting most chance k-mer hits. To increase sensitivity, MMseqs2 not only compares the original k-mer extracted from the query to the database k-mers, but also generates a list of similar k-mers, where letters within the query k-mer are progressively replaced with similar characters according to a substitution matrix. To reduce dependencies between consecutive k-mers in sequences, we use spaced k-mers, where the extracted substring contains k informative positions and number of ignored positions.

The MMseqs2 prefilter is thus a heuristic that quickly identifies likely homologous sequence pairs, while rejecting few real hits and allowing few false hits. This does not only result in high sensitivity, but also, crucially, a large speedup over the slower downstream algorithms, as they have to investigate a much-reduced search space.

Alignment. MMseqs2 uses a Farrar-style Smith-Waterman-Gotoh [Farrar, 2007, Zhao et al., 2013] algorithm to identify homologous hits. This algorithm is vectorized to use Single Instruction Multiple Data (SIMD) processing capabilities of modern CPUs. SIMD-vectorization allows efficient processing of multiple values with a single CPU instruction.

DIAMOND

DIAMOND [Buchfink et al., 2015, 2021] is another fast and sensitive sequence aligner. It drives the homology-based search for the assignment of taxonomic labels in CAT [Von Meijenfeldt et al., 2019], thus also requiring some details about its inner workings.

In contrast to MMseqs2, DIAMOND precomputes sorted lists of k-mers for both query and target sequences. To identify shared k-mers between query and target sequences, both lists are linearly read together. This approach results in a high utilization of caches within modern CPUs and, thus, in fast and efficient identification of likely homologous pairs. To increase sensitivity, DIAMOND repeats this search with multiple spaced k-mer patterns. The found pairs are then aligned with a SIMD accelerated Smith-Waterman algorithm to identify homologous hits.

1.4 Metagenomics and the sequencing revolution

Since the Human Genome Project first decoded the human genome, sequencing technology has improved dramatically in throughput and price [Venter et al., 2001]. An estimated 500 million $ to 1 billion $ were spent to produce the first high-quality human reference genome [Wetterstrand, 2021a]. Illumina, a biotechnology company specialized in sequencing machines, claimed in 2014 to have reached the stated goal of the 1000$ genome. However, the National Human Genome Research Institute (NHGRI) states that this price point has only become widely accessible in the beginning of 2019 [Wetterstrand, 2021b]. Figure 1.7 shows the drop in costs for sequencing a megabase of DNA since the human genome project.

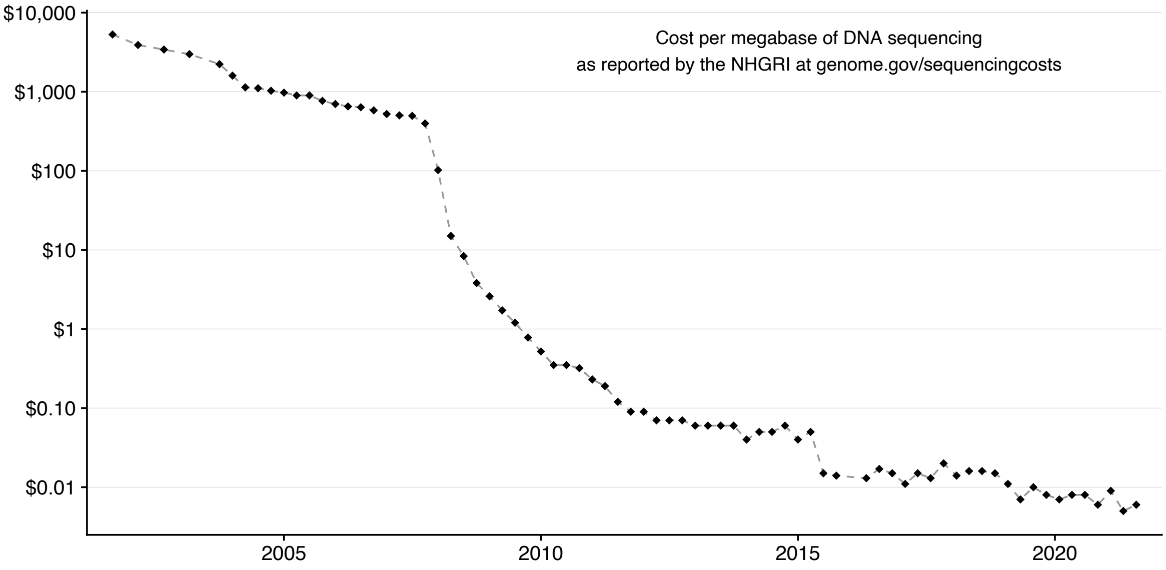


Figure 1.7: Costs to sequence one megabase of DNA according to the NHGRI.

The availability of fast and affordable sequencing technologies has enabled a revolution of sequencing samples directly from various environmental sources, such as soils, oceans, animal guts and many more. The study of microorganisms directly from their native environments without culturing in a laboratory has been named metagenomics [Handelsman et al., 1998].

Early metagenomics experiments could only resolve few genomes from low diversity sources (e.g., five bacterial genomes from acid mine drainage by Tyson et al. [2004]). Technology and software have improved to the point that it is now possible to resolve hundreds of thousands of high-quality genomes (e.g. Almeida et al. [2021]) or billions of proteins [Steinegger et al., 2019b, Mitchell et al., 2020] from metagenomic experiments.

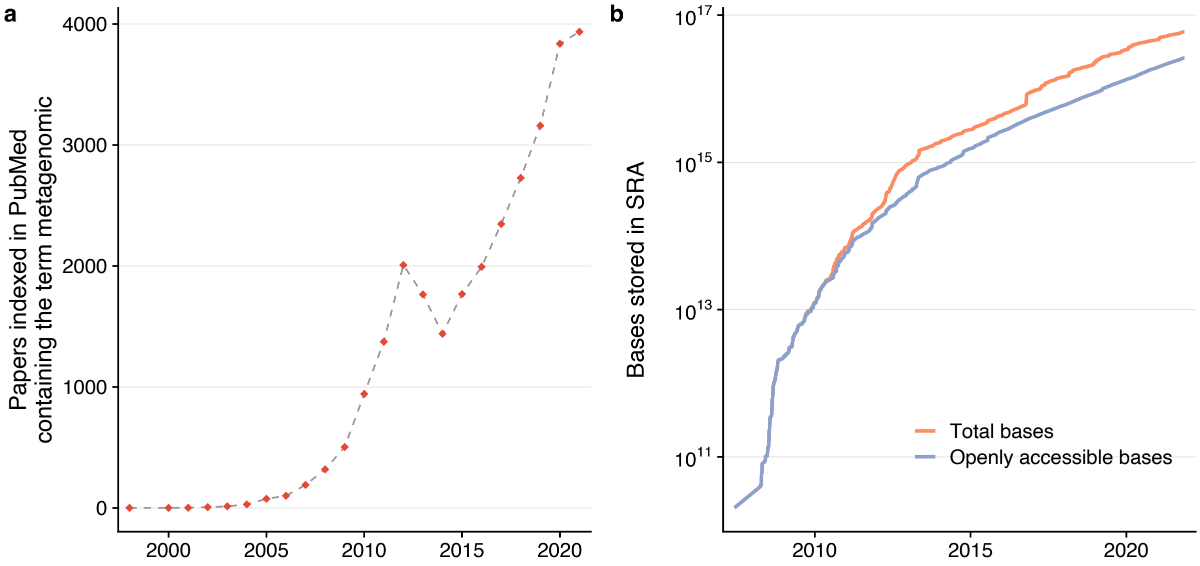


Figure 1.8: (a) Growth of the number of publications indexed in PubMed [Sayers et al., 2021a] containing the term "metagenomic". (b) Growth of the SRA in log-scale.

The results of sequencing experiments are deposited – often by mandate if they are to be published – in the Sequence Read Archive (SRA) and made available to the scientific community. Over 53 PB of data in a total of 12.1 million indexed sequencing experiments are now deposited in the SRA, a figure that continues to grow rapidly (Fig. 1.8 right). As metadata is not standardized [Kasmanas et al., 2021] estimating the share of metagenomic data proves itself difficult. Searching the SRA for the term "metagenomic" currently (Nov 2021) results in over 2 million hits, a substantial fraction of all data sets.

1.5 Taxonomy

Taxonomic classification of species began as a modern science with Carl Linnaeus, who is also called the "father of modern taxonomy" [Calisher, 2007]. Linnaeus introduced a hierarchical system of taxonomic ranks – from the root of all life to phyla, classes, orders, families, genera and species – and the binomial nomenclature with two-part scientific names for species. The first, generic name identifies the genus and the second, the specific names, identifies the species.

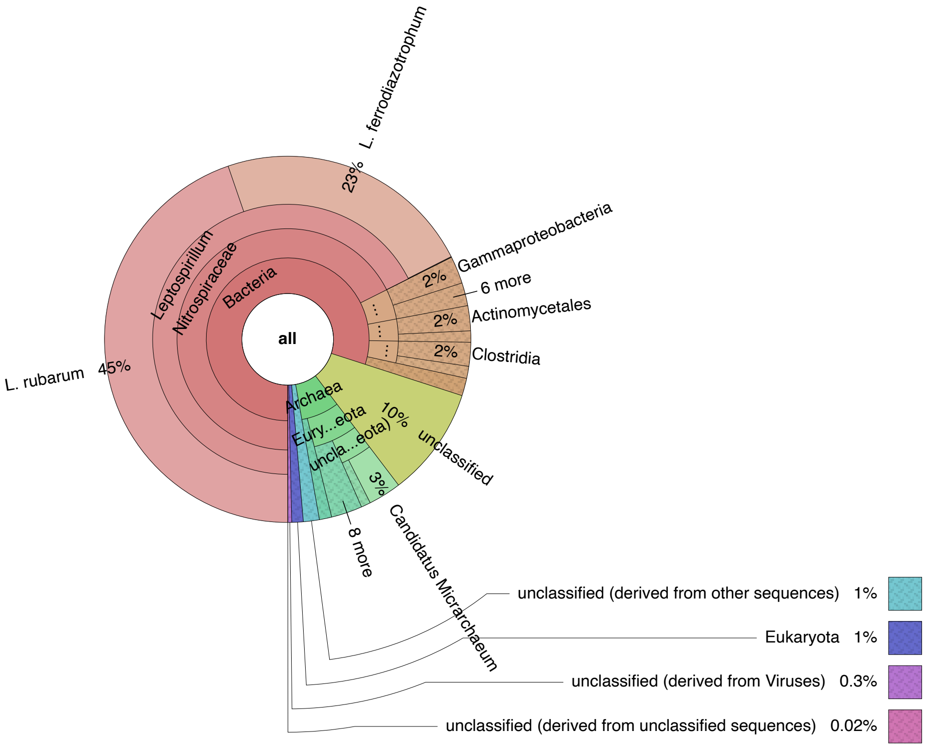


Figure 1.9: Taxonomic visualization of the acid mine drainage dataset [Tyson et al., 2004] mentioned in section 1.4. Created with Krona [Ondov et al., 2011].

Today, the actual naming is governed by international societies, such as the International Committee on Taxonomy of Viruses (ICVT) for viruses (among many others). A central repository for taxonomic classification is hosted by the National Center for Biotechnology Information (NCBI) [Federhen, 2012], which maintains a digitally readable taxonomy.

Historically, the exact definition of a species has been controversial [Hey et al., 2005]. The concept of reproductive isolation, defining species in terms of the ability to produce reproductively viable offspring, has been successfully applied to differentiate most sexually reproducing eukaryotic species. The vast majority of microorganisms (such as viruses, bacteria, protists etc.) however, do not reproduce sexually, or only rarely. Additionally, horizontal gene transfer – the exchange of genetic material between individual microorganisms – further complicates matters.

For bacteria, a 95% average nucleotide identity of the 16S ribosomal RNA subunit is commonly used as a cutoff to separate species [Goris et al., 2007]. Additionally, bacterial species have to be culturable and samples have to be deposited into public repositories before they are assigned an official name.

Most species identified in metagenomic experiments are however unculturable. Thus, the diversity of microorganisms found in these is absent from taxonomic databases. A proposed solution to allow naming of unculturable microbes was to allow genome sequences to be directly deposited in the public repositories [Murray et al., 2020]. However, this proposal was rejected.

To alleviate this, independent taxonomies based on genomic data alone were developed. The Genome Taxonomy Database (GTDB) [Parks et al., 2018, 2021] contains a consistent taxonomy based on 258 406 bacterial and archaeal genomes. These genomes originate from isolates, metagenome assembled genomes and single amplified genomes, covering a much broader range of microbial life.

Inconsistencies between these approaches remain an important challenge. A widely known example of the difficulty in classifying bacterial species are the two bacteria *Escherichia coli* and *Shigella flexneri*. Both species have highly similar genomes, thus should – from a biological perspective – reside in the same genus [Zuo et al., 2013]. However, in clinical care infections by *E. coli* and *S. flexneri* have vastly different risks and treatments [Devanga Ragupathi et al., 2018]. The decision of the GTDB to resolve the biological misclassification by moving *S. flexneri* into the Escherichia genus and renaming it to *Escherichia flexneri* was met with heavy criticism [Sanford et al., 2021].

Determining the taxonomic identity of unknown sequences is an important task in sequence analysis. The most commonly used technique is to transfer taxonomical labels to unknown sequences by comparing them to a reference database annotated with taxonomic labels. As reference database searches rarely yield only one exact match of an unknown sequence, taxonomic labels are assigned by combining evidence from multiple close hits and computing e.g., a lowest common ancestor as the best possible label.

A distinction can be made between methods using nucleotide- and protein reference databases for taxonomic assignment. Since protein sequences are more conserved than their DNA counterparts, protein alignments can be used to detect homology into the *twilight zone* of 20–35% sequence identity [Rost, 1999]. With this strategy, taxonomic labels can be assigned to sequences, which do not have sufficient similarity to any known sequence on the DNA level.

Many computational methods are available for fast taxonomic classification (as reviewed in Sczyrba et al. [2017], Ye et al. [2019], Meyer et al. [2021]). Kraken [Wood and Salzberg, 2014, Wood et al., 2019] is a commonly used method for taxonomic assignment of short sequence reads. Methods like Kaiju [Menzel et al., 2016] and Megan [Huson et al., 2007] are commonly used to exploit protein databases. Fewer methods taxonomically annotate assembled contigs; one such method is CAT [Von Meijenfeldt et al., 2019] and another is MMseqs2 taxonomy [Mirdita et al., 2021], presented here.

1.6 Protein structure prediction

Early experiments have shown that denatured (forcibly unfolded) proteins will often regain their function when returned to normal physiological conditions [Anson, 1945]. As a consequence it was postulated that there must exist a unique mapping of the amino acid sequence to its native three-dimensional structure [Anfinsen, 1973] (Anfinsen's dogma). Even though some broad classes of exceptions are now known (e.g., proteins with multiple native conformations [Fox et al., 1986] or intrinsically disordered proteins [Dunker et al., 2013]), the dogma can still be considered to broadly hold; the amino acid sequence of a protein largely determines its three-dimensional structure.

An implication of Anfinsen's dogma is that it may be possible to predict the three-dimensional structure given an amino acid sequence. The main task lies in predicting the positions of the protein backbone, the chain of α-carbons of all the amino acids. The backbone is usually encoded in terms of three relevant angles (φ,ψ,ω) that uniquely define the position of each α-carbon in three-dimensional space (Ramachandran et al. [1963], also called dihedral angles). This results in a combinatorial explosion of possible arrangements even for relatively short amino acid chains, making it infeasible to solve the protein folding problem with algorithms that rely on exploring the full parameter space.

The interest in producing *in silico* predicted protein structures is not solely academic. Experimental structure determination methods (i.e., X-ray crystallography, NMR spectroscopy and cryogenic electron microscopy) are costly in terms of procuring the expensive machines, continuously maintaining these and, perhaps most important, in terms of the laboriously and error-prone preparation of each protein whose structure should be solved.

Luckily, proteins are real biological entities and fold under biochemical and physical constraints. Thus, the space of possible arrangements is limited and researchers began to investigate how to exploit these limits to predict protein structures computationally [Levitt and Warshel, 1975].

The Critical Assessment of Structure Prediction (CASP) competition was started to objectively monitor the progress of the state-of-the-art of computational protein structure determination [Moult et al., 1995]. In almost three decades of biennial competitions, the research community has investigated various approaches.

In the following I want to highlight broad classes of methods that found prominence in CASP and conclude with a brief introduction to AlphaFold2, which has been particularly groundbreaking.

Traditionally, protein structure prediction was tackled through template-based-modeling or (template-)free modeling [Kuhlman and Bradley, 2019]. However, this distinction has become much more blurred with the rise of machine learning based methods [AlQuraishi, 2021a].

Template-based modeling. In template-based modeling, homology search is used to find sequences of already solved structures or solved fragments thereof – also called templates. As protein structures are much more conserved than their sequences [Illergård et al., 2009], structure information can be transferred from the template to the target to predict the structure of the unknown sequence. Particularly, software that is able to detect very remote homology to a known structure is useful to increase the coverage of solved parts for structure determination [Hildebrand et al., 2009].

Template-free modeling. When no homologous templates can be found, much more difficult template-free modeling approaches have to be used, as no information from already solved structures is available. A widely used strategy is to search for residue pairs that are distant from each other in the amino acid sequence but close in three-dimensional space (under 8 Å). If enough such pairs can be found, then the possible conformation-space could be restricted enough to make protein structure predictions feasible [Kim et al., 2014]. One way to find such residue pairs is based on the observation that structural constraints lead to a characteristic co-evolution footprint in the MSA that can be exploited to improve structure prediction. Many methods for *contact predictions* were developed [Kamisetty et al., 2013, Seemayer et al., 2014, Jones et al., 2015, Wang et al., 2018] and their application to protein structure prediction led to significant improvements in CASP12 [Schaarschmidt et al., 2018].

Molecular dynamics and model refinement. The seemingly most natural computational solution to the protein folding problem is the full physical *ab initio* simulation of the folding of a protein within its environment. This approach is also called molecular dynamics. However, this is extremely computationally demanding and only possible for short proteins and timescales [Bonneau and Baker, 2001]. Thus, molecular dynamics has remained confined to *model refinement*, where it is used to improve models predicted by faster methods [Heo and Feig, 2018a]. Despite the many efforts, progress has stalled in recent CASP competitions in relation to other methods [Heo and Feig, 2018b].

AlphaFold. AlphaFold1 [Senior et al., 2020] and AlphaFold2 [Jumper et al., 2021a] won the CASP13 and CASP14 competitions, respectively. AlphaFold2's predictions in CASP14 achieved a median GDT-TS of 92.4% [Jumper et al., 2021b, Kryshtafovych et al., 2021] – a value comparable to the error of margin of experimental structure determination methods. This success is attributed to the extensive expertise of the authors in implementing and training novel machine learning methods based on end-to-end deep learning [Glasmachers, 2017, AlQuraishi, 2021b]. In end-to-end deep learning, the problem – from input features to output structure prediction – is encoded as a single neural network.

While AlphaFold1 employs residual networks [He et al., 2015] to predict dihedral angles and inter-residue distances from a contact map and PSSM input, AlphaFold2 leverages transformer neural networks architecture [Vaswani et al., 2017] to directly predict structures from MSA and template input. A remarkable feature of AlphaFold2 is that it often produces high quality structures with very few diverse sequences provided as input MSA features ($\sim$30).

2 MMseqs2 desktop and local web server app for fast, interactive sequence searches

Publication:

MMseqs2 desktop and local web server app for fast, interactive sequence searches

M. Mirdita, M. Steinegger, J. Söding[†]

(†) corresponding author

Bioinformatics (2019): 35(16), 2856–2858.
Cited 41 times since 01/2019.

2.1 Author contributions

M.M., M.S. & J.S. performed research and wrote the manuscript. **M.M.** & M.S. implemented features in MMseqs2. **M.M.** implemented the remaining software.

2.2 Code and data availability

The MMseqs2 app is available as free open source software (GPLv3) at `app.mmseqs.com`. An example webserver is available at `search.mmseqs.com`.

Bioinformatics, 35(16), 2019, 2856–2858
doi: 10.1093/bioinformatics/bty1057
Advance Access Publication Date: 7 January 2019
Applications Note

Sequence analysis

MMseqs2 desktop and local web server app for fast, interactive sequence searches

Milot Mirdita[1], **Martin Steinegger**[1,2,*] **and Johannes Söding**[1,*]

[1]Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, 37077 Göttingen, Germany and [2]Department of Chemistry, Seoul National University, Seoul 08826, Korea

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on September 3, 2018; revised on November 17, 2018; editorial decision on December 6, 2018; accepted on January 4, 2019

Abstract

Summary: The MMseqs2 desktop and web server app facilitates interactive sequence searches through custom protein sequence and profile databases on personal workstations. By eliminating MMseqs2's runtime overhead, we reduced response times to a few seconds at sensitivities close to BLAST.

Availability and implementation: The app is easy to install for non-experts. GPLv3-licensed code, pre-built desktop app packages for Windows, MacOS and Linux, Docker images for the web server application and a demo web server are available at https://search.mmseqs.com.

Contact: martin.steinegger@mpibpc.mpg.de or soeding@mpibpc.mpg.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The most popular sequence similarity search tool, BLAST (Altschul *et al.*, 1990, 1997), has garnered ~7000 citations per year during the last 5 years, attesting to the unremitting importance of sequence searches for biology. This popularity may be largely owed to the excellent web services with short response times despite fast-growing databases provided by the NCBI/NIH, which requires a huge compute infrastructure. The distributed approach of running searches *locally* on personal computers or IT platforms of companies and research groups allows for custom databases, high availability and protects sensitive data. But web server applications for local homology searches are slow as they mostly rely on BLAST (e.g. Deng *et al.*, 2007; Priyam *et al.*, 2015). Here, we present an application software to search with protein and nucleotide sequences through custom protein sequence and profile databases using MMseqs2 (Steinegger and Söding, 2017), achieving response times of seconds instead of minutes at a similar sensitivity as BLAST.

2 Materials and methods

2.1 Reduced runtime overhead

MMseqs2 owes its sensitivity and speed mainly to its pre-filtering stage, which rejects ~99.99% of sequences. The pre-filter uses a reverse *k*-mer index table for the target database and also requires matrices with similarity scores between 2-mers and between 3-mers to generate the lists of similar 7-mers (Steinegger and Söding, 2017). Reading in the index table and computing these matrices on-the-fly takes ~0.5 min of runtime overhead for each search. We reduced this to 0.05 s by (1) writing the index table, the matrices and other pre-computable data into a file if it does not yet exist, memory mapping the file to take advantage of the system page cache (for detailed memory requirements see Supplementary Materials) and (3) optimizing I/O operations.

2.2 Optimized sequence-to-profile search mode

The index table for profile databases stores, for each position in a profile, all *k*-mers with a profile similarity score above a threshold set by `-s`. The number of similar *k*-mers grows exponentially with *k*. To save memory, we chose a short $k = 5$ as default for this mode. We also added to Mmseqs2 utilities for creating profiles from multiple sequence alignments (MSAs) and converting between profile formats.

2.3 Desktop and web server app

Based on the same code base, the application can be either deployed through Docker containers to be accessed through web browsers or

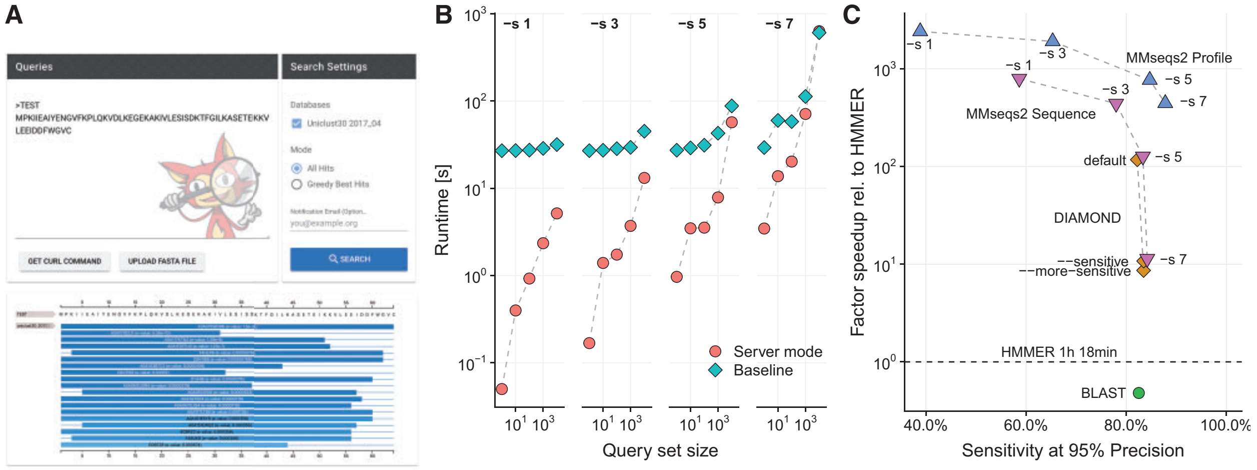


Fig. 1. (**A**) Screenshots of the search interface and result visualization. (**B**) Runtime of searches with the baseline MMseqs2 (square) and the new server mode (circle) at four sensitivity settings (`-s`). (**C**) Domain annotation: Speedup versus sensitivity at 95% precision for MMseqs2 (triangle: sequence-profile search, upside-down triangle: sequence–sequence search; sensitivity settings: `-s 1, 3, 5, 7`), DIAMOND (square; *default*, `-sensitive`, `-more-sensitive`) and BLAST (circle). HMMER3 matches to Pfam domains are used as ground truth. The speed-ups exclude the times to format the databases

packaged as a desktop GUI application with the Electron framework (electronjs.org). In either case, the backend part of the application provides a RESTful API and worker scheduling. The server supports protein, translated nucleotide and nucleotide sequence searches and iterative and reverse profile searches.

The application takes a list of either protein or nucleotide sequences in FASTA/FASTQ format as query input. To generate a target search database, the application takes a FASTA/FASTQ file for protein sequence searches or a STOCKHOLM MSA file for protein profile searches. Search results are shown with a customized feature-viewer (github.com/calipho-sib/feature-viewer) (Fig. 1A) and can be downloaded in tabular BLAST format.

3 Results

Figure 1B demonstrates the reduction of runtime overhead by comparing the runtimes of the Mmseqs2 version without ('baseline') to the new version with pre-computations and memory mapping ('server mode'). Runtimes refer to searches with amino acid query sets of 1, 10, 100, 1000 and 10 000 sequences of average length 350 (sampled from the Uniclust30 database) through the Uniclust30 2017_10 database (Mirdita *et al.*, 2017) with 13.5 million sequences, measured on a server with 2 Intel Xeon E5-2680 v4 CPUs with 14 cores each. The index table and matrix pre-computation (~3 min 40 s) is not included in the runtimes.

To test the quality and speed of annotating Pfam domains on genes assembled from metagenomics data, we built a test set by sampling 100 000 full-length sequences longer than 150 residues from our Marine Eukaryotic Reference Catalogue (Steinegger *et al.*, 2018), clustering this set to 30% maximum pairwise sequence identity with MMseqs2 and sampling 10 000 sequences from the redundancy-reduced set. We annotated these sequences with PfamA 31.0 domains (Finn *et al.*, 2014) using HMMER3 (Finn *et al.*, 2011).

We then compared how well the sequence-sequence searches of MMseqs2, BLAST and DIAMOND (Buchfink *et al.*, 2015) and the sequence-to-profile searches of MMseqs2 could find the correct domain annotations. For the sequence-sequence search methods, we built a database from all sequences in `PfamA.full` MSAs and reported as E-value of a Pfam domain the E-value for the best-matching sequence from its MSA. We defined a search as true positive (TP) if the top match was annotated by HMMER3 with an E-value better than 10^{-3} and as false positive (FP) if the top match was not annotated with an HMMER3 E-value below 1. All other searches were considered ambiguous and ignored. For each method, we determined the E-value at which the precision TP/(TP+FP) is 95% and measured the sensitivity at that E-value.

As Figure 1C shows, MMseqs2 sequence-to-profile searches are ~30 times faster than sequence-sequence searches with DIAMOND, MMseqs2 and BLAST and ~300 times faster than HMMER3. MMseqs2 sequence-to-profile searches reach 87% relative sensitivity at 95% precision, making them an attractive alternative to HMMER3 when speed is critical.

4 Conclusion

The desktop and web server app for MMseqs2 performs fast sequence searches at unprecedented speed-to-sensitivity trade-off on local computers. Thousand queries take only a minute to search through fifteen million sequences of the Uniclust30 database, much faster than NCBI's BLAST website. We hope the MMseqs2 app will also empower users unfamiliar with command line interfaces to perform fast and sensitive searches with their own sequence and profile databases.

Acknowledgements

The authors thank Yuna Kwon for crafting the 'little Marv' mascot.

Funding

This work was supported by the European Research Council in the framework of its Horizon 2020 Framework Programme for Research and Innovation (grant 'Virus-X', project no. 685778).

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Buchfink,B. *et al.* (2015) Fast and sensitive protein alignment using DIAMOND. *Nat. Methods*, **12**, 59–60.

Deng,W. *et al.* (2007) Viroblast: a stand-alone blast web server for flexible queries of multiple databases and user's datasets. *Bioinformatics*, **23**, 2334–2336.

Finn,R.D. *et al.* (2011) HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.*, **39**, W29–W37.

Finn,R.D. *et al.* (2014) Pfam: the protein families database. *Nucleic Acids Res.*, **42**, D222–D230.

Mirdita,M. *et al.* (2017) Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Res.*, **45**, D170–D176.

Priyam,A. *et al.* (2015) Sequenceserver: a modern graphical user interface for custom BLAST databases. *bioRxiv, 033142*.

Steinegger,M. and Söding,J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.

Steinegger,M. *et al.* (2018) Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *bioRxiv, 386110*.

Supplementary Material for MMseqs2 desktop and local web server app for fast, interactive sequence searches

Mirdita M.,[1] Steinegger M.,[1, 2] and Söding J.[1]

[1] *Quantitative and Computational Biology Group,*
Max Planck Institute for Biophysical Chemistry, Göttingen, Germany
[2] *Department of Chemistry, Seoul National University, Seoul, Korea*

I. MEMORY REQUIREMENTS

The MMseqs2 web server keeps the precomputed index fully in main memory using memory mapping with the `mmap` system call.

For optimal speed, the whole precomputed index file has to reside in the operating system's page cache.

The precomputed index consists predominantly of two parts: (1) the k-mer lookup table and (2) the residues. The memory consumption grows linearly with the number of residues in the database.

A. Sequence-sequence search requirements

The following formula can be used to estimate the size M of the precomputed index file in the case of a sequence-sequence search.

$$\begin{aligned} M &\approx 8\,\mathrm{b} \times N \times L \\ &+ 8\,\mathrm{b} \times a^k \\ &+ 32\,\mathrm{b} \times N \end{aligned}$$

Where N is the database size, L is the average sequence length, a the alphabet size (typically 20, with the unknown residue X excluded) and k the k-mer size.

The following table shows memory requirements for a few example databases of different sizes:

Name	Release	Entries	Size $k = 6$	Size $k = 7$
SwissProt	2018_08	558125	2.3GB	12GB
Uniclust30	2018_08	30M	56GB	64GB
Uniclust50	2018_08	45M	96GB	105GB
Uniclust90	2018_08	120M	295GB	304GB

TABLE I. Memory requirements for typical sequence search databases.

B. Sequence-profile search requirements

The target profile search keeps all similar k-mers for each profile in memory. The memory consumption M is dominated by the average k-mer list length (K_{avg}) per profile column. K_{avg} depends on the chosen sensitivity setting, higher sensitivity results in longer k-mer lists.

$$M \approx 6\,\mathrm{b} \times N \times L_p \times K_{\mathrm{avg}}$$

Where N is the database size, L_p is the average profile column length.

The following table shows memory requirements and typical K_{avg} values for the Pfam-A profile database at different sensitivity settings:

Sensitivity	$k = 5$	K_{avg}	$k = 6$	K_{avg}
$s = 1$	123MB	4.7	1.1GB	19.2
$s = 3$	633MB	26	5.0GB	190
$s = 5$	5.3GB	231	25GB	1070
$s = 7$	26GB	1401	119G	5279

TABLE II. Memory requirements for Pfam-A 31.0 profile search databases (16479 profiles, 4006517 total residues in consensus sequences).

II. SOFTWARE VERSIONS

Name	Version
MMseqs2	Git: 8a8520c
blastp	2.6.0+
hmmer	3.1b2
diamond	0.9.19

TABLE III. Software versions used in this manuscript.

3 Fast and sensitive taxonomic assignment to metagenomic contigs

Publication:

Fast and sensitive taxonomic assignment to metagenomic contigs

M. Mirdita, M. Steinegger, F. Breitwieser, J. Söding†, E. Levy Karin†

(†) corresponding author

Bioinformatics (2021): 37(**18**), 3029-3031.
Cited 13 times since 03/2021.

3.1 Author contributions

M.M., J.S. & E.L.K. performed research and wrote the manuscript. **M.M.**, M.S., F.B. and E.L.K. implemented features in MMseqs2. **M.M.** and E.L.K. implemented the taxonomy assignment algorithm.

3.2 Code and software availability

MMseqs2 taxonomy is available as free open source software (GPLv3) as part of MMseqs2 at `mmseqs.com`.

Bioinformatics, 2021, 1–3
doi: 10.1093/bioinformatics/btab184
Advance Access Publication Date: 18 March 2021
Applications Note

Sequence analysis

Fast and sensitive taxonomic assignment to metagenomic contigs

M. Mirdita [1], M. Steinegger [2,3,4], F. Breitwieser [5], J. Söding [1,6,]* and E. Levy Karin [1,]*

[1]Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany, [2]School of Biological Sciences, Seoul National University, Seoul, South Korea, [3]Institute of Molecular Biology and Genetics, Seoul National University, Seoul, South Korea, [4]Artificial Intelligence Institute, Seoul National University, Seoul, South Korea, [5]Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, MD 21205, USA and [6]Campus-Institut Data Science (CIDAS), Göttingen, Germany

*To whom correspondence should be addressed.

Associate Editor: Janet Kelso

Received on November 16, 2020; revised on February 26, 2021; editorial decision on March 11, 2021; accepted on March 16, 2021

Abstract

Summary: MMseqs2 taxonomy is a new tool to assign taxonomic labels to metagenomic contigs. It extracts all possible protein fragments from each contig, quickly retains those that can contribute to taxonomic annotation, assigns them with robust labels and determines the contig's taxonomic identity by weighted voting. Its fragment extraction step is suitable for the analysis of all domains of life. MMseqs2 taxonomy is 2–18× faster than state-of-the-art tools and also contains new modules for creating and manipulating taxonomic reference databases as well as reporting and visualizing taxonomic assignments.

Availability and implementation: MMseqs2 taxonomy is part of the MMseqs2 free open-source software package available for Linux, macOS and Windows at https://mmseqs.com.

Contact: soeding@mpibpc.mpg.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Metagenomic studies shine a light on previously unstudied parts of the tree of life. However, unraveling taxonomic composition accurately and quickly remains a challenge. While most methods label short metagenomic reads (reviewed in Sczyrba *et al.*, 2017), only a handful (e.g. Huson *et al.*, 2018) assign entire contigs, even though this should lead to improved accuracy.

Recently, von Meijenfeldt *et al.* (2019) developed CAT, a tool for taxonomic annotation of contigs based on protein homologies to a reference database. It combines Prodigal (Hyatt *et al.*, 2010) for predicting open reading frames (ORFs), DIAMOND (Buchfink *et al.*, 2015) to search with the translated ORFs, and logic to aggregate individual ORF annotations. CAT achieved higher precision than state-of-the-art tools on bacterial benchmarks. Despite its advantage over existing methods, CAT has limitations: (i) Prodigal was designed for prokaryotes and not eukaryotes (West *et al.*, 2018); (ii) Prodigal runs single-threaded, limiting applicability to metagenomics; (iii) CAT's *r* parameter determines the cut-off score below each ORF's top-hit above which hits are included in the ORF's lowest common ancestor (LCA) computation. Although the authors provide guidelines to set *r*, it is unclear how general they are.

Here, we present MMseqs2 taxonomy, a novel protein-search-based tool for taxonomy assignment to contigs. It overcomes the aforementioned limitations by extracting all possible protein fragments, covering the coding repertoire of all domains of life. It quickly eliminates fragments that do not bear minimal similarity to the reference database, and searches with the remaining ones. MMseqs2 taxonomy uses an approximate 2bLCA (Hingamp *et al.*, 2013) strategy to assign translated fragments to taxonomic nodes (Supplementary Material). The hits for the approximate 2bLCA computation are determined automatically, saving the need to tune an equivalent of CAT's *r* parameter. It outperforms CAT on bacterial and eukaryotic datasets.

2 Materials and methods

Input. Contigs are provided as (compressed) FASTA/Q files. As reference, the *databases* workflow can download and prepare various public taxonomy databases, such as, nr (Agarwala *et al.*, 2018), UniProt (Bateman, 2019) or GTDB (Parks *et al.*, 2020). Alternatively, users can prepare their own taxonomic reference database (see MMseqs2 wiki).

Algorithm. The four main steps are described in Figure 1A.

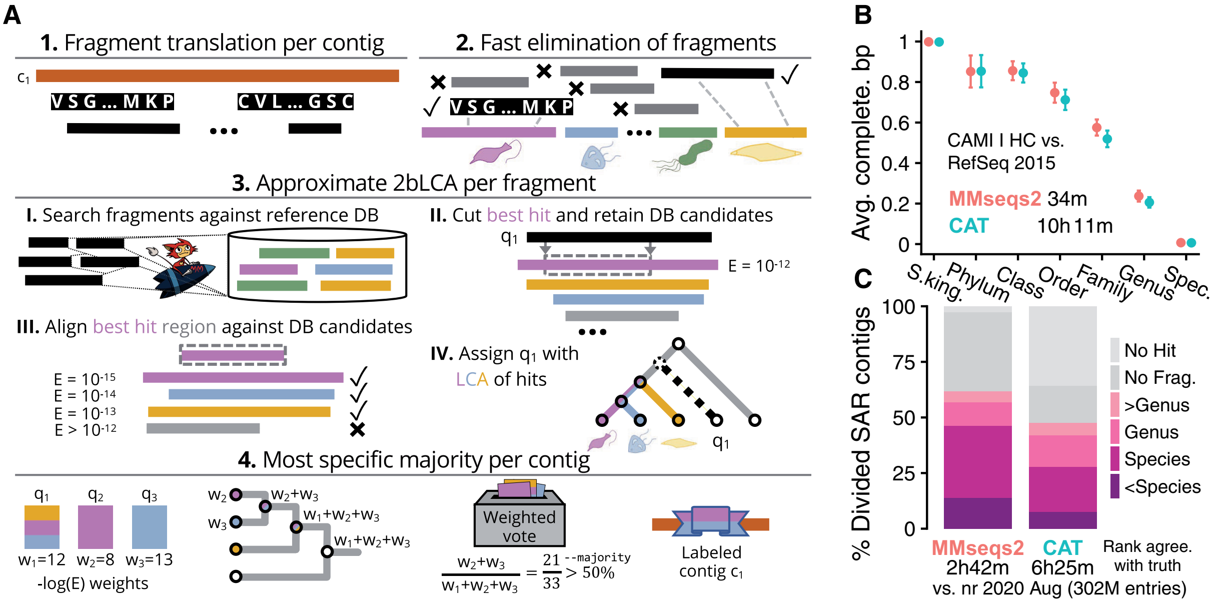

A
1. Fragment translation per contig
c₁
V S G ... M K P
C V L ... G S C
2. Fast elimination of fragments
V S G ... M K P
3. Approximate 2bLCA per fragment
I. Search fragments against reference DB
II. Cut best hit and retain DB candidates
q₁
E = 10⁻¹²
III. Align best hit region against DB candidates
E = 10⁻¹⁵
E = 10⁻¹⁴
E = 10⁻¹³
E > 10⁻¹²
IV. Assign q₁ with LCA of hits
q₁
4. Most specific majority per contig
q₁ q₂ q₃
w₁=12 w₂=8 w₃=13
-log(E) weights
w₂ w₂+w₃
w₃ w₁+w₂+w₃
w₁+w₂+w₃
Weighted vote
(w₂+w₃)/(w₁+w₂+w₃) = 21/33 > 50%
--majority
Labeled contig c₁
B
Avg. complete. bp
1
0.8
0.6
0.4
0.2
0
CAMI I HC vs. RefSeq 2015
MMseqs2 34m
CAT 10h11m
S.king. Phylum Class Order Family Genus Spec.
C
% Divided SAR contigs
100
75
50
25
0
No Hit
No Frag.
>Genus
Genus
Species
<Species
MMseqs2 2h42m
CAT 6h25m
Rank agree. with truth
vs. nr 2020 Aug (302M entries)

Fig. 1. (**A**) Taxonomy assignment algorithm in four steps: (1) Translate all possible protein fragments in six frames from all contigs. (2) Reject fragments unlikely to find a taxonomic hit in later stages (full details in Supplementary Material). (3) Assign taxonomic nodes using an approximate 2bLCA procedure. Each query fragment q is searched against the reference database, resulting in a list l of all its homologous targets. The aligned region between q and the best hit t [with E-value $E(q, t)$] is aligned against all targets in l. Assign q the LCA of the taxonomic lables of all target sequences that have an E-value lower than $E(q, t)$. Realigning l allows avoiding the costly second search of 2bLCA. (4) Each assigned q contributes its weight ($-log\ E(q, t)$) to its taxonomic label and all labels above it, up to the root. The contig's taxonomic node is determined as the most specific taxonomic label, which has a support of at least the −−majority parameter. The support of a label is the sum of its contributing weights divided by the total sum of weights. (**B**) MMseqs2 taxonomy (red) is ∼18× faster and achieves similar average completeness to CAT (turquoise) on a bacterial benchmark. (**C**) MMseqs2 assigns taxa to eukaryotic SAR contigs more accurately than CAT across all phylogenetic levels, at twice the speed. At species level, MMseqs2 taxonomy classifies 46% contigs correctly versus 28% for CAT. Runtimes measured on a 2×14-core Intel E5-2680v4 server with 768 GB RAM

Output. MMseqs2 taxonomy returns the following eight fields for each contig accession: (i) the taxonomic identifier (taxid) of the assigned label, (ii) rank, (iii) name, followed by the number of fragments: (iv) retained, (v) taxonomically assigned, and (vi) in agreement with the contig label (i.e. same taxid or have it as an ancestor), (vii) the support the taxid received and, optionally, (viii) the full lineage. The result can be converted to a TSV-file, and to a Kraken (Wood *et al.*, 2019) report or a Krona (Ondov *et al.*, 2011) visualization (Supplementary Material).

3 Results

Bacterial dataset. The CAMI-I high-complexity challenge and its accompanying RefSeq 2015 reference database (Sczyrba *et al.*, 2017) were given to MMseqs2 and CAT. AMBER v2 (Meyer *et al.*, 2018) was used to assess the taxonomic assignment by computing the average completeness (Fig. 1B) and purity (Supplementary Fig. S1) bp using its taxonomic binning benchmark mode. At similar assignment quality, MMseqs2 taxonomy is 18× faster than CAT. Using the nr, MMseqs2 is 10× faster (Supplementary Fig. S2).

Eukaryotic dataset. All 57 SAR (taxid 2698737) RefSeq assemblies and their taxonomic labels were downloaded from NCBI in 08/2020. To resemble metagenomic data, their scaffolds were randomly divided following the length distribution of contigs assembled for sample ERR873969 of eukaryotic Tara Oceans (Carradec *et al.*, 2018), resulting in 2.7 million non-overlapping contigs with a minimal length of 300 bp. Using nr from 08/2020, MMseqs2 classified more contigs than CAT (62% versus 47%). For 36%, CAT extracted a fragment that did not hit the reference, suggesting fragments extracted by MMseqs2 are more informative for eukaryotic taxonomic annotation (Fig. 1C, Supplementary Fig. S3).

4 Conclusion

MMseqs2 taxonomy is as accurate as CAT on a bacterial dataset while being 3–18× faster and requiring fewer parameters. Its extracted fragments make it suitable for analyzing eukaryotes. It is accompanied by several taxonomy utility modules to assist with taxonomic analyses.

Funding

E.L.K. is a FEBS long-term fellowship recipient and an EMBO nonstipendiary long-term fellow. The work was supported by the BMBF CompLifeSci project horizontal4meta; the ERC's Horizon 2020 Framework Programme ['Virus-X', project no. 685778]; the National Research Foundation of Korea grant funded by the Korean government (MEST) [2019R1A6A1A10073437, NRF-2020M3A9G7103933]; and the Creative-Pioneering Researchers Program through Seoul National University.

Conflict of Interest: none declared.

Data availability

The data used to benchmark MMseqs2 taxonomy in this study are openly available from https://data.cami-challenge.org/participate at the Databases section. The SAR assemblies were downloaded from NCBI in 08/2020 and processed as described.

References

Agarwala,R. *et al.* (2018) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, **46**, D8–D13.

Bateman,A. (2019) UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.*, **47**, D506–D515.

Buchfink,B. *et al.* (2015) Fast and sensitive protein alignment using DIAMOND. *Nat. Methods*, **12**, 59–60.

Carradec,Q. *et al.*, Tara Oceans Coordinators. (2018) A global ocean atlas of eukaryotic genes. *Nat. Commun.*, **9**, 373.

Hingamp,P. *et al.* (2013) Exploring nucleo-cytoplasmic large DNA viruses in Tara Oceans microbial metagenomes. *ISME J.*, **7**, 1678–1695.

Huson,D.H. *et al.* (2018) MEGAN-LR: new algorithms allow accurate binning and easy interactive exploration of metagenomic long reads and contigs. *Biol. Direct.*, **13**, 6.

Hyatt,D. *et al.* (2010) Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinform.*, **11**, 119.

Meyer,F. *et al.* (2018) AMBER: Assessment of Metagenome BinnERs. *Gigascience*, **7**, giy069.

Ondov,B.D. *et al.* (2011) Interactive metagenomic visualization in a Web browser. *BMC Bioinform.*, **12**, 385.

Parks,D.H. *et al.* (2020) A complete domain-to-species taxonomy for Bacteria and Archaea. *Nat. Biotechnol.*, **38**, 1079–1086.

Sczyrba,A. *et al.* (2017) Critical assessment of metagenome interpretation—a benchmark of metagenomics software. *Nat. Methods*, **14**, 1063–1071.

von Meijenfeldt,F.A.B. *et al.* (2019) Robust taxonomic classification of uncharted microbial sequences and bins with CAT and BAT. *Genome Biol.*, **20**, 217.

West,P.T. *et al.* (2018) Genome-reconstruction for eukaryotes from complex natural microbial communities. *Genome Res.*, **28**, 569–580.

Wood,D.E. *et al.* (2019) Improved metagenomic analysis with Kraken 2. *Genome Biol.*, **20**, 257.

Supplementary Material for Fast and sensitive taxonomic assignment to metagenomic contigs using MMseqs2

Mirdita M.,[1] Steinegger M.,[2,3,4] Breitwieser F.,[5] Söding J.,[1,6] and Levy Karin E.[1]

[1] *Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany*
[2] *School of Biological Sciences, Seoul National University, Seoul, South Korea*
[3] *Institute of Molecular Biology and Genetics, Seoul National University, Seoul, South Korea*
[4] *Artificial Intelligence Institute, Seoul National University, Seoul, South Korea*
[5] *Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, USA*
[6] *Campus-Institut Data Science (CIDAS), Göttingen, Germany*

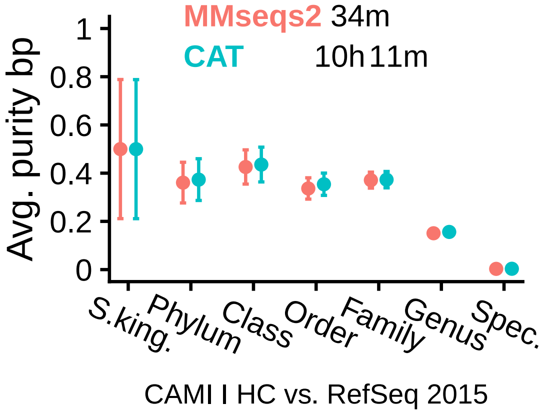


FIG. S1. MMseqs2 (red) is ∼18x faster and achieves similar average purity bp to CAT (turquoise) on a bacterial benchmark. The CAMI taxonomic reference database does not contain any species, which were included in the challenge. All classifiers score 0% bp completeness/purity at that rank. Runtimes measured on a server with 2x14-core Intel E5-2680v4 CPUs and 768GB RAM.

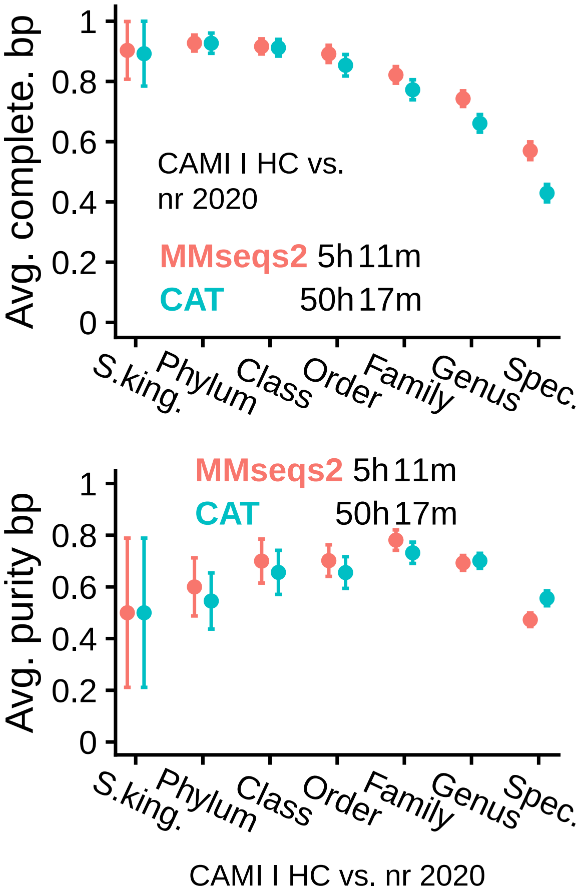


FIG. S2. MMseqs2 (red) is ∼10x faster and achieves similar accuracy to CAT (turquoise) on a bacterial benchmark, using the nr as reference (16x more entries than RefSeq 2015). In contrast to Fig 1B and Fig S1, the nr database does not exclude any species, which allows for correct classifications at the species level. All runtimes were measured on a server with two 14-core Intel E5-2680v4 CPUs and 768GB RAM.

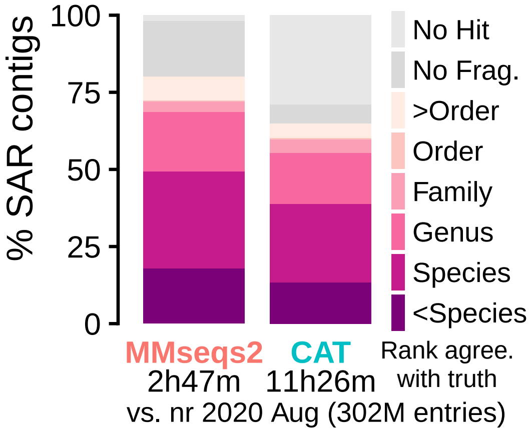


FIG. S3. The fragments extracted and retained by MMseqs2 from 66,630 eukaryotic scaffolds result in more correctly classified scaffolds than those extracted for CAT by Prodigal. All runtimes were measured on a server with two 14-core Intel E5-2680v4 CPUs and 768GB RAM.

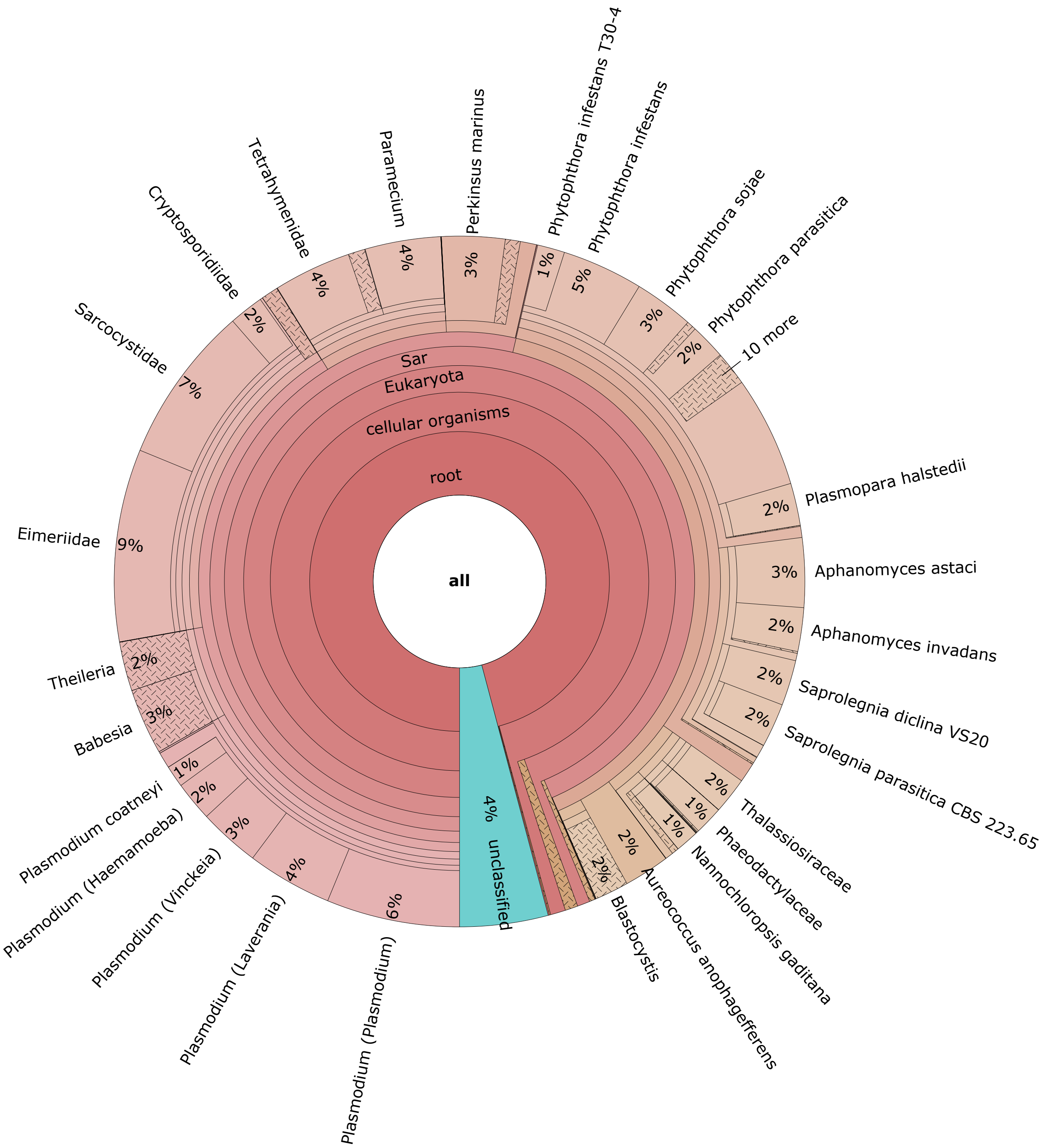


FIG. S4. Krona visualization of the classified contigs of Fig. 1C (62% of all contigs) generated using `mmseqs taxonomyreport --report-mode 1`.

EARLY PROTEIN FRAGMENT REJECTION

MMseqs2 taxonomy uses the `prefilter` module to find the translated fragment-to-reference match with the highest number of consecutive similar k-mer matches on the same diagonal. Fragments with fewer than three matches to any reference sequence are removed. The remaining pairs are aligned without gaps using `rescorediagonal` and the fragment in each pair is retained if the pair's E-value is smaller than 100.

APPROXIMATE 2BLCA

The 2bLCA procedure consists of two searches: (I) A search with a query sequence against a set of target sequences. (II) A search with the aligned region of the most significant sequence match against the same target sequences. The taxonomic labels of all hits with an E-value smaller or equal to the best hit E-value in search I are used to compute an LCA.

The prefilter of MMseqs2 can quickly identify candidates of homology, which are then verified by a costly alignment step. We approximate 2bLCA by assuming that most candidates found in a prefiltering step of search I would also cover the candidates found by search II. Thus, we reuse the same list of prefiltering candidates for both alignment steps.

Additionally, we exploit MMseqs2's support for multiple alignment modes to calculate only the score and E-value, or to additionally compute the alignment boundaries. In the initial alignment of the query against the target candidates, we use the first mode to find the hit with the best E-value and then we recompute, for the best hit only, the alignment boundaries with the slower, second alignment mode. To compute the E-values of the best matching aligned region to the target sequences we also use the first, faster alignment mode.

We applied `--max-accept 30` and `--max-rejected 5` in the first alignment and `--max-rejected 5` in the second alignment, to further speed up the alignments.

Name	Version	Comment
MMseqs2	Git: 7da33b0	Benchmarks
MMseqs2	Git: 6379422	nr DB creation
CAT	v5.1.1	

TABLE S1. Software versions used in this manuscript.

Sequence set	Version	Entries	Residues/Bases
CAMI I HC GSA	2015	42k	2.8B nucl.
CAMI I RefSeq	2015	16M	6.5B aa.
SAR (unchopped)	08/2020	67k	2.2B nucl.
SAR (chopped)	08/2020	2.7M	2.2B nucl.
nr	08/2020	303M	109B aa.

TABLE S2. Sequence sets used in this manuscript.

Method	Target DB	Peak RAM
MMseqs2	CAMI I RefSeq	60 GB
MMseqs2	nr	253 GB
CAT	CAMI I RefSeq	45 GB
CAT	nr	63 GB

TABLE S3. Peak RAM use of MMseqs2 and CAT with CAMI I HC dataset. Memory use was measured on a server with two 14-core Intel E5-2680v4 CPUs and 768GB RAM. Note, however, that both methods can split the database into chunks and search them one after the other to adapt to available system memory.

4 ColabFold - Making protein folding accessible to all

Publication:

ColabFold - Making protein folding accessible to all

M. Mirdita†, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov†, M. Steinegger†

(†) corresponding authors and equal contributors are ordered alphabetically

bioRxiv (2021), in review.
Cited 45 times since 08/2021.

4.1 Author contributions

M.M., K.S., S.O. and M.S. performed research and programming, **M.M.**, S.O. and M.S. jointly designed the research and wrote the manuscript. Y.M. provided the initial methodology for hetero-complex modeling and created an installer for use on local servers. L.H. provided initial benchmarking.

4.2 Code and software availability

ColabFold is free open-source software (MIT) available at `github.com/sokrypton/ColabFold`. Its novel environmental databases are available at `colabfold.mmseqs.com`.

ColabFold - Making protein folding accessible to all

Milot Mirdita,[1, *] Konstantin Schütze,[2] Yoshitaka Moriwaki,[3, 4] Lim Heo,[5] Sergey Ovchinnikov,[6, 7, *] and Martin Steinegger[2, 8, *]

ColabFold offers accelerated protein structure and complex predictions by combining the fast homology search of MMseqs2 with AlphaFold2 or RoseTTAFold. ColabFold's $20-30$x faster search and optimized model use allows predicting thousands of proteins per day on a server with one GPU. Coupled with Google Colaboratory, ColabFold becomes a free and accessible platform for protein folding. ColabFold is open-source software available at github.com/sokrypton/ColabFold. Its novel environmental databases are available at colabfold.mmseqs.com

Contact: milot.mirdita@mpibpc.mpg.de, so@fas.harvard.edu, martin.steinegger@snu.ac.kr

Predicting the three-dimensional structure of a protein from its sequence alone remains an unsolved problem. However, by exploiting the information in multiple sequence alignments (MSAs) of related proteins as raw input features for end-to-end training, AlphaFold2 [1] was able to predict the 3D atomic coordinates of folded protein structures at an median GDT-TS of 92.4% in the latest CASP14 [2] competition. The accuracy of many of the predicted structures was within the error margin of experimental structure determination methods. Many ideas of AlphaFold2 were independently reproduced and implemented in RoseTTAFold [3]. Additionally to single chain predictions, RoseTTAFold was shown to model protein complexes. Evans *et al.* [3] also announced a refined version of AlphaFold2 for complex prediction. Thus, two highly accurate open-source prediction methods are now publicly available.

In order to leverage the power of these methods researchers require powerful compute-capabilities. First, to build diverse MSAs, large collections of protein sequences from public reference [4] and environmental [1, 5] databases are searched using the most sensitive homology detection methods HMMer [6] and HHblits [7]. Due to the large database sizes these searches can take up to hours for a single protein, while requiring over two terabyte of storage space alone. Second, to execute the deep neural networks GPUs with a large amount of GPU RAM are required even for relatively common protein sizes of ∼1000 residues. Though, for these the MSA generation dominates the overall run-time (**Supplementary Fig. 1**).

To enable researchers without these resources to use AlphaFold2 independent solutions based on Google Colaboratory were developed. Colaboratory is a proprietary version of Jupyter Notebook hosted by Google. It is accessible for free to logged-in users and includes access to powerful GPUs. Tunyasuvunakool *et al.* [8] developed an AlphaFold2 Jupyter Notebook for Google Colaboratory (referred to as AlphaFold-Colab), where the input MSA is built by searching with HMMer against a clustered UniProt and an eight-fold reduced environmental databases. Resulting in less accurate predictions, while still requiring long search times.

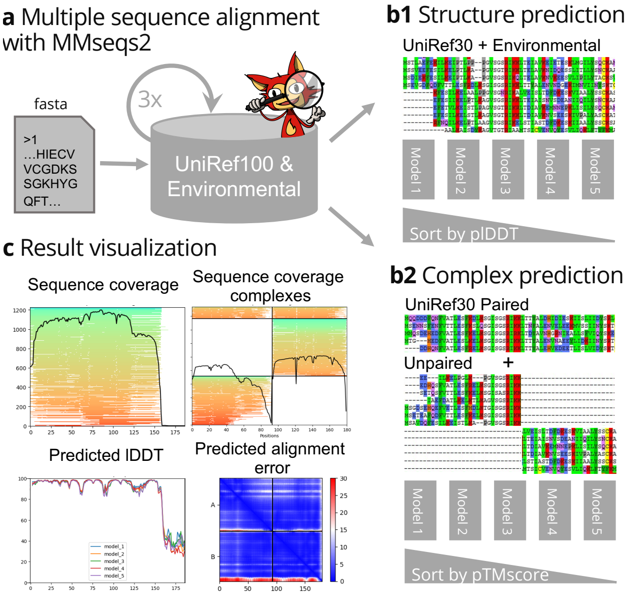


FIG. 1. (**a**) ColabFold sends a FASTA input sequence to a MMseqs2 server searching two databases UniRef100 and a database of environmental sequences with three profile-search iterations each. The second database is searched using a sequence-profile generated from the UniRef100 search as input. The server generates two MSAs in A3M format containing all detected sequences. (**b1**) For single structure predictions we filter both A3Ms using a diversity aware filter and return this to be provided as the MSA input feature to the AlphaFold2 models. (**b2**) For complex prediction we pair the top hits within the same species to resolve the inter-complex contacts and additionally add two unpaired MSAs (same to **b1**) to guide the structure prediction. (**c**) To help researchers judge the prediction quality we visualize MSA depth and diversity and show the AlphaFold2 confidence measures (pLDDT and PAE).

Here, we present ColabFold, a fast and easy to use software for protein structure and homo- and heteromer complex prediction, for use as a Jupyter Notebook inside Google Colaboratory, on researchers' local computers as a notebook or through a command line interface. ColabFold speed-ups the prediction by replacing the AlphaFold2's input feature generation stage with a fast MMseqs2 [9, 10] search. It additionally implements speed-ups for predictions of multiple structures by avoiding recompilation and adding early stop criteria. We show that ColabFold outperforms AlphaFold-Colab and matches AlphaFold2 on CASP14 targets while being 20-30 times faster. ColabFold can compute a proteome (excluding proteins >1000 residues) in 41 hours on a consumer GPU.

[1] Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany. [2] School of Biological Sciences, Seoul National University, Seoul, South Korea. [3] Department of Biotechnology, Graduate School of Agricultural and Life Sciences, The University of Tokyo, Tokyo, Japan. [4] Collaborative Research Institute for Innovative Microbiology, The University of Tokyo, Tokyo, Japan. [5] Department of Biochemistry and Molecular Biology, Michigan State University, East Lansing, MI 48824, USA. [6] JHDSF Program, Harvard University, Cambridge, MA 02138, USA. [7] FAS Division of Science, Harvard University, Cambridge, MA 02138, USA. [8] Artificial Intelligence Institute, Seoul National University, Seoul, South Korea [*] These authors contributed equally and are ordered alphabetically.

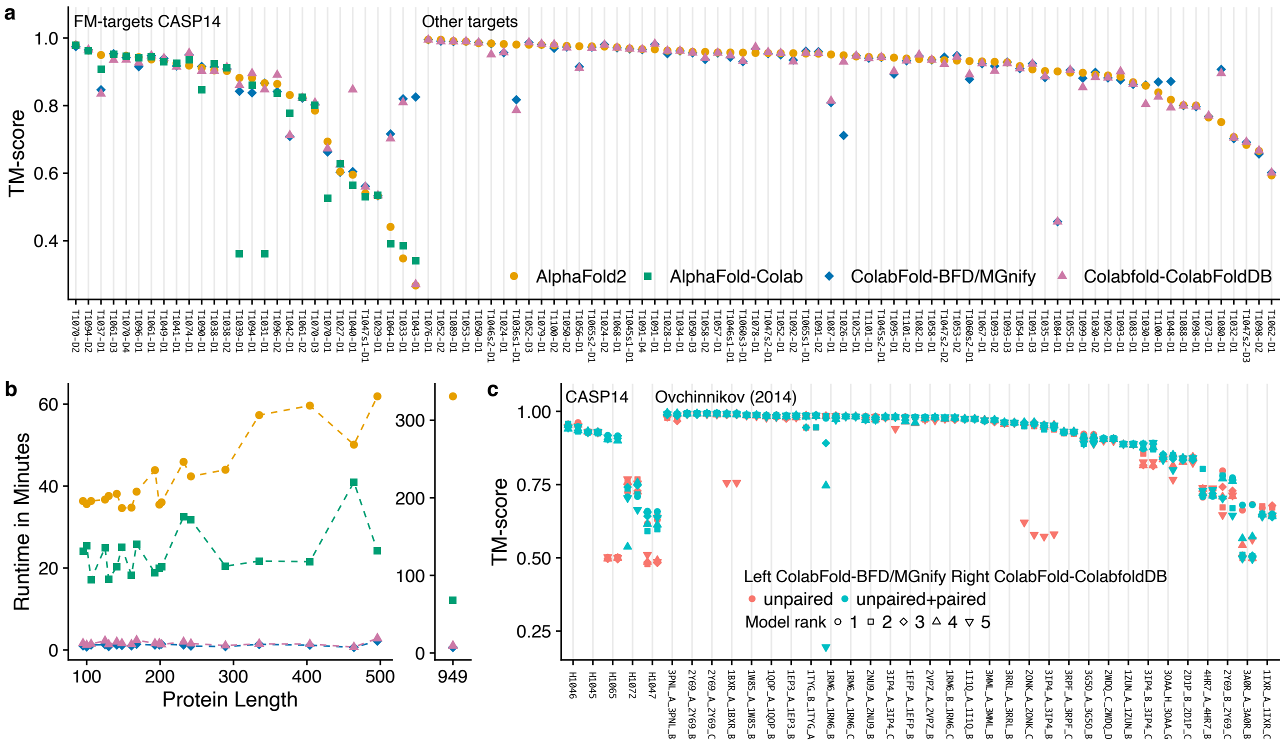


FIG. 2. (**a**) Structure prediction comparison of AlphaFold2 (yellow), AlphFold-Colab (green) and ColabFold with BFD/MGnify (blue) and with the ColabFoldDB (magenta) using predictions of 96 domains of 69 CASP14 targets. The 28 domains from the 20 free-modeling (FM) targets are shown first. FM targets were used to optimize MMseqs2 search parameters. Each target was evaluated for each individual domain (in total 96 domains). (**b**) MSA generation time for each CASP14 FM target sorted by protein length (same colors as before). FM target T1064 shown separately to improve readability. (**c**) Comparison of ColabFold complex predictions with unpaired (red) and unpaired+paired (blue) MSA-pairing modes, the databases BFD/MGnify (left of line) and ColabFoldDB (right). See **Supplementary Fig. 2** for comparison to paired-only mode.

ColabFold (**Fig. 1**) consists of three parts: (1) An MMseqs2 based homology search server to build diverse MSAs and to find templates. The server efficiently aligns input sequence(s) against the UniRef100, the PDB70 and an environmental sequence set. (2) A Python library that communicates with the MMseqs2 search server, prepares the input features for (single or complex) structure inference, and visualizes of results. This library also implements a command line interface. (3) Jupyter notebooks for basic, advanced and batch use (Methods "ColabFold notebooks") using the Python library.

In ColabFold we replace the sensitive search methods HMMer and HHblits by MMseqs2. We optimized the MSA generation by MMseqs2 to have the following three properties: (1) MSA generation should be fast. (2) The MSA has to capture diversity well and (3) it has to be small enough to run on GPUs with limited RAM. Reducing the memory requirement is especially helpful in Google Colaboratory where the provided GPU is selected from a pool with widely differing capabilities. While (1) is achieved through the fast MMseqs2 prefilter for (2 and 3) we developed a search workflow to maximize sensitivity (Methods "MSA generation") and a new filter that samples the sequence space evenly (Methods "New diversity aware filter" and **Supplementary Fig. 3**). Prediction quality highly depends on the input MSA. However, often only a few (∼30) sufficiently diverse sequences are enough to produce high quality predictions [1].

Additionally, we combined the BFD and MGnify databases that are used in AlphaFold2 by HHblits and HMMer respectively into a combined redundancy reduced version we refer to as BFD/MGnify (Methods "Reducing size of BFD/MGnify"). The environmental search database presented an opportunity to improve structure predictions of non-bacterial sequences, as e.g., eukaryotic protein diversity is not well represented in the BFD and MGnify databases. Limitations in assembly and gene calling due to complex intron/exon structures result in under representation in reference databases. We therefore extended the BFD/MGnify with additional metagenomic protein catalogues containing eukaryotic proteins [11, 12, 13], phage catalogues [14, 15] and an updated version of MetaClust [16]. We refer to this database as ColabFoldDB (Methods "ColabFoldDB"). In **Supplementary Fig. 4** we show that the ColabFoldDB in comparison to the BFD/MGnify produces more diverse MSAs for PFAM [17] domains with < 30 members.

To compare the accuracy of predicted structures we compared AlphaFold2 (default settings with templates), AlphaFold-Colab (no templates), and ColabFold (no templates) with the BFD/MGnify and ColabFoldDB on TM-scores for all targets from the CASP14 competition (**Fig. 2a**), split by free modeling (FM) targets on the left and the remaining ones on the right. We show this split as we used the FM-targets for optimization of search workflow parameters.

The mean TM-scores for the FM targets are 0.826, 0.818, 0.79 and 0.744 for ColabFold (BFD/MGnify), ColabFold (ColabFoldDB), AlphaFold2 and the AlphaFold-Colab, respectively. Over all CASP14 targets the TM-scores are 0.88, 0.877 and 0.88 for the former three respectively. For AlphaFold-Colab we measured TM-scores only for FM targets as it cannot be used stand-alone.

ColabFold could not predict T1084 well as MMseqs2 suppresses all databases hits as false positives due to its amino acid composition filter and masking procedure. If these filters are deactivated T1084 can be predicted with an TM-score of 0.872 (**Supplementary Fig. 5**).

ColabFold is on average 5x faster for single predictions than AlphaFold2 and AlphaFold-Colab, when taking both MSA generation (**Fig. 2b**) and model inference into account.

AlphaFold2 itself has no capabilities to model complexes. However, we found that by combining two sequences with a glycine linker [18] it could often successfully model complexes. Shortly afterwards, Baek [19] found that incrementing the model-internal residue index - the method that was used in RoseTTAFold - could also be used in AlphaFold2.

For high quality predictions it was shown that sequences should be provided in paired-form to AlphaFold2 [20]. We implemented a similar pairing procedure (Methods "MSA pairing for complex prediction") and show the complex prediction capabilities of ColabFold in **Fig. 2c**. We achieve high accuracy in complex prediction in two datasets from Ovchinnikov *et al.* [21] and the CASP14 protein complex targets with two unique sequences (Methods "Complex Benchmark" for benchmark details). We note though that the structures from [21] were already public and were likely used as individual chains during the training of AlphaFold2.

Fig. 3 shows two examples of ColabFold's complex prediction capabilities: (**a**) shows a homo-six-mer and (**b**) shows a D-methionine transport system composed of three different proteins. For single structure prediction AlphaFold2 provides a pLDDT measure to indicate the prediction quality. A high pLDDT does not necessarily indicate a correct complex prediction, though the inter-complex predicted alignment error (PAE) helps to rank complexes. We visualize plots of PAE and complex conformation to help users judge the prediction quality of a complex. An example for heteromer complex prediction is shown in **Supplementary Fig. 6** with its PAE plot. Furthermore, ColabFold complexes were successfully used to aid the cryo-EM structure determination of the 120 MDa human nucleopore complex [22].

In ColabFold we expose many internal parameters of AlphaFold2 to aid users to model difficult targets, such as the recycle count (default 3). It controls the number of times the prediction is repeatedly feed through the model. For difficult targets as well as for designed proteins without known homologs additional recycling iterations can result in a high quality prediction (**Supplementary Fig. 7**).

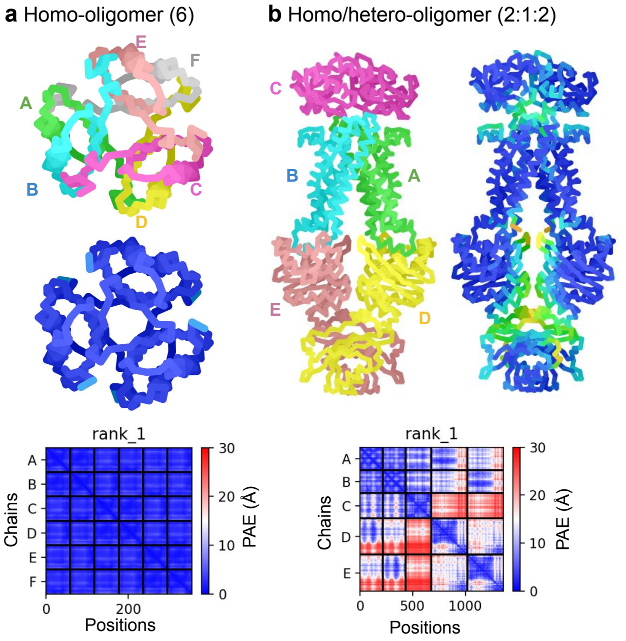


FIG. 3. Anecdotal examples showcasing the capabilities of advanced ColabFold features. (**a**) Setting the homo-oligomer setting to 6, allows modeling of the homo-6-mer structure of 4-Oxalocrotonate Tautomerase. Colored by chain (top), pLDDT (predicted Local Distance Difference Test, bottom). The inter PAE (Predicted Aligned Error) between chains is very low indicating a confident prediction. (**b**) Providing three different proteins with 2:1:2 homo-oligomer setting allows modeling a hetero-complex with mismatching symmetries of the D-methionine transport system.

To meet the demand for high throughput structure prediction we introduced several features in ColabFold. (1) MSA generation can be executed in batch-mode independently from model batch-inference. (2) We compile only two of the five AlphaFold2 models and reuse weights. (3) We provide a batch execution mode, that avoids recompilation for sequences of similar length. (4) We implement early stop criteria, to avoid running additional recycles or models if a sufficiently accurate structure was already found. All together, we show that the proteome of 1762 proteins shorter than 1000 aa of the archaeon *Methanocaldococcus jannaschii* can be predicted in 40h on one Nvidia RTX 3090 (Methods "Proteome Benchmark").

ColabFold builds beyond the initial offerings of Alphafold2 by improving its sequence search, providing tools for modeling homo- and heteromer complexes, exposing advanced functionality, expanding the environmental databases and performing structure prediction in batch within a minute.

In summary, ColabFold makes high quality protein structure prediction accessible and additionally provides novel features to explore the full potential of AlphaFold2 and RoseTTAFold.

REFERENCES

[1] Jumper, J. *et al.* *Nature* **596**, 583–589 (2021).
[2] Kryshtafovych, A. *et al.* *Proteins* 1–11 (2021).
[3] Evans, R. *et al.* *bioRxiv* 2021.10.04.463034 (2021).
[4] UniProt Consortium. *Nucleic Acids Res.* **47**, D506–D515 (2019).
[5] Mitchell, A. L. *et al.* *Nucleic Acids Res.* **48**, D570–D578 (2020).
[6] Eddy, S. R. *PLoS Comput. Biol.* **7**, e1002195 (2011).
[7] Steinegger, M. *et al.* *BMC Bioinform.* **20**, 473 (2019).
[8] Tunyasuvunakool, K. *et al.* *Nature* **596**, 590–596 (2021).
[9] Steinegger, M. & Söding, J. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
[10] Mirdita, M. *et al.* *Bioinformatics* **35**, 2856–2858 (2019).
[11] Levy Karin, E. *et al.* *Microbiome* **8**, 48 (2020).
[12] Delmont, T. O. *et al.* *bioRxiv* 2020.10.15.341214 (2020).
[13] Alexander, H. *et al.* *bioRxiv* 2021.07.25.453713 (2021).
[14] Nayfach, S. *et al.* *Nat. Microbiol.* **6**, 960–970 (2021).
[15] Camarillo-Guerrero, L. F. *et al.* *Cell* **184**, 1098–1109.e9 (2021).
[16] Steinegger, M. & Söding, J. *Nat. Commun.* **9**, 2542 (2018).
[17] Mistry, J. *et al.* *Nucleic Acids Res.* **49** (2021).
[18] Moriwaki, Y. AlphaFold2 can also predict heterocomplexes. all you have to do is input the two sequences you want to predict and connect them with a long linker. https://twitter.com/Ag_smith/status/1417063635000598528 (2021).
[19] Baek, M. Adding a big enough number for "residue_index" feature is enough to model hetero-complex using AlphaFold (green&cyan: crystal structure / magenta: predicted model w/ residue_index modification). https://twitter.com/minkbaek/status/1417538291709071362 (2021).
[20] Bryant, P. *et al.* *bioRxiv* 2021.09.15.460468 (2021).
[21] Ovchinnikov, S. *et al.* *eLife* **3**, e02030 (2014).
[22] Mosalaganti, S. *et al.* *bioRxiv* 2021.10.26.465776 (2021).

ACKNOWLEDGEMENTS

We thank Johannes Söding for providing computational resources. John Jumper and Tim Green for answering questions regarding AF2. Minkyung Baek for the complex residue trick. Do-Yoon Kim for creating the ColabFold logo. Enzo Guerrero-Araya and Jakub Kaczmarzyk for providing bug fixes. Alon Markovich and Julia Varga for notifying us about MSA quality issues. Harriet Alexander for providing the TOPAZ proteins as a single file to download. We thank all users for using ColabFold and reporting issues.

This work used the Scientific Compute Cluster at GWDG, the joint data center of Max Planck Society for the Advancement of Science (MPG) and University of Göttingen. Milot Mirdita acknowledges the BMBF CompLifeSci project horizontal4meta. Martin Steinegger acknowledges support from the National Research Foundation of Korea grant [2019R1A6A1A10073437, 2020M3A9G7103933, 2021R1C1C102065]; New Faculty Startup Fund and the Creative-Pioneering Researchers Program through Seoul National University. Yoshitaka Moriwaki acknowledges support from Platform Project for Supporting Drug Discovery and Life Science Research (Basis for Supporting Innovative Drug Discovery and Life Science Research (BINDS)) from AMED under Grant Number JP21am0101107. For this project, Sergey Ovchinnikov was supported by the National Science Foundation under Grant No. MCB2032259. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

AUTHOR CONTRIBUTION

M.M., K.S., S.O. and M.S. performed research and programming, M.M., S.O. and M.S. jointly designed the research and wrote the manuscript. Y.M. provided the initial methodology for hetero-complex modeling and created an installer for use on local servers. L.H. provided initial benchmarking.

COMPETING INTERESTS

The authors declare no competing interests.

MATERIALS AND METHODS

ColabFold notebooks ColabFold has four main Jupyter notebooks [23]: `AlphaFold2_mmseqs2` for basic use that supports protein structure prediction using (1) MSAs generated by MMseqs2, (2) custom MSA upload, (3) using template information, (4) relaxing the predicted structures using amber force fields [24], and (5) monomer complex prediction. `AlphaFold2_advanced` for advanced users additionally supports (6) MSA generation using HMMer (same as AlphaFold-Colab), (7) the sampling of diverse structures by iterating through a series of random seeds (`num_samples`), and (8) control of AlphaFold2 model internals, such as changing the number of recycles (`max_recycle`), number of ensembles (`num_ensemble`), and enabling the stochastic part of the models via the (`is_training`) option. `AlphaFold2_batch` for batch prediction of multiple sequences or MSAs. The batch notebook saves time by avoiding recompilation of the AlphaFold2 models ("Avoid recompiling during batch computation") for each individual input sequence. `RoseTTAFold` for basic use of RoseTTAFold that supports protein structure prediction using (1) MSAs generated by MMseqs2, (2) custom MSAs and (4) sidechain prediction using SCWRL4 [25].

ColabFold command line interface We initially focused on making ColabFold as widely available as possible through our Notebooks running in Google Colaboratory. To meet the demand for a version that runs on local users' machines, we released "LocalColabFold". LocalColabFold can take command line arguments to specify an input FASTA file, an output directory, and various options to tweak structure predictions. LocalColabFold runs on wide range of operating systems, such as Windows 10 or later (using Windows Subsystem for Linux 2), macOS, and Linux. The structure inference and energy minimization are accelerated if a CUDA 11.1 or later compatible GPU is present. LocalColabFold is available as free open-source software at `github.com/YoshitakaMo/localcolabfold`.

Specifically for running large numbers of protein complexes or structure predictions e.g., for an entire proteome (Methods "Proteome benchmark"), we provide the `colabfold_batch` command line tool through the `colabfold` python package. It can be installed with `pip install colabfold`, followed by `pip install -U "jax[cuda]" -f https://storage.googleapis.com/jax-releases/jax_releases.html`. It can be used as `colabfold_batch input_file_or_directory output_directory`, supporting FASTA, A3M and CSV files as input.

MSA generation by MMseqs2 ColabFold sends the query sequence to a MMseqs2 server [12]. It searches the sequence(s) with three iterations against the consensus sequences of the UniRef30, a clustered version of the UniRef100 [26]. We accept hits with an E-value of lower than 0.1. For each hit, we realign its respective UniRef100 cluster member using the profile generated by the last iterative search, filter them (Methods "New diversity aware filter") and add these to the MSA. This expanding search results in a speed up of ∼10x as only 29.3 million cluster consensus sequence are searched instead of all 277.5 million UniRef100 sequences. Additionally, it has the advantages to be more sensitive since the cluster consensus sequences are used. We use the UniRef30 sequence-profile to perform an iterative search against the BFD/MGnify or ColabFoldDB using the same parameters, filters and expansion strategy.

New diversity aware filter To limit the number of hits in the final MSA we use the HHblits diversity filtering algorithm [8] implemented in MMseqs2 in multiple stages: (1) During UniRef cluster expansion, we filter each individual UniRef30 cluster before adding the cluster members to the MSA, such that no cluster-pair has a higher maximum sequence identity than 95% (`--max-seq-id 0.95`. (2) After realignment enable only the `--qsc 0.8` threshold and disable all other thresholds (`--qid 0 --diff 0 --max-seq-id 1.0`). Additionally, the qsc filtering is only used if least 100 hits were found (`--filter-min-enable 100`). (3) During MSA construction we filter again with the following parameters: `--filter-min-enable 1000 --diff 3000 --qid 0.0,0.2,0.4,0.6,0.8,1.0 --qsc 0 --max-seq-id 0.95`. Here, we extended the HHblits filtering algorithm to filter within a given sequence identity bucket, such that it cannot eliminate redundancy across filter buckets. Our filter keeps the 3000 most diverse sequences in the identity buckets]0.0-0.2],]0.2-0.4],]0.4-0.6],]0.6-0.8] and]0.8-1.0]. In buckets containing less than 1000 hits we disable the filtering.

New MMseqs2 pre-computed index to support expanding cluster members MMseqs2 was initially built to perform fast many-against-many sequence searches. Mirdita *et al.* [11] improved it to also support fast single-against-many searches. This type of search requires the database to be index and stored in memory. `mmseqs createindex` indexes the sequences and stores all time-consuming-to-compute data structures used for MMseqs2 searches to disk. We load the index into the operating systems cache using *vmtouch* (`github.com/hoytech/vmtouch`) to allow calls to the different MMseqs2 modules become near-overhead free. We extended the index to store, in addition to the already present cluster consensus sequences, all member sequences and the pairwise alignments of the cluster representatives to the cluster members. With these resident in cache, we eliminate the overhead of the remaining module calls.

Reducing size of BFD/MGnify To keep all required sequences and data structures in memory we needed to reduce the size of the environmental databases BFD and MGnify, as both databases together would have required ∼517 GB RAM for headers and sequences alone.

BFD is a clustered protein database consisting of ∼2.2 billion proteins organized in 64 million clusters. MGnify (2019_05) contains ∼300 million environmental proteins. We merged both databases by searching the MGnify sequences against the BFD cluster representative sequences using MMseqs2. Each MGnify sequence with a sequence identity of >30% and a local alignment that covers at least 90% of its length is assigned to the respective BFD cluster. All unassigned sequences are clustered at 30% sequence identity and 90% coverage (`--min-seq-id 0.3 -c 0.3 --cov-mode 1 -s 3`) and merged with the BFD clusters, resulting in 182 million clusters. In order to reduce the size of the database we filtered each cluster keeping only the 10 most diverse sequences using (`mmseqs filterresult --diff 10`). This reduced the total number of sequences from 2.5 billion to 513 million, thus requiring only 84 GB RAM for headers and sequences.

ColabFoldDB We built ColabFoldDB by expanding the BFD/MGnify with metagenomic sequences from various environments. To update the database, we searched the proteins from the SMAG (eukaryotes) [14], MetaEuk (eukaryotes) [13], TOPAZ (eukaryotes) [15], MGV (DNA viruses) [16], GPD (bacteriophages) [17] and updated version of MetaClust [17] against the BFD/MGnify centriods using MMseqs2 and assigned each sequence to the respective cluster if they have a 30% sequence identity at a 90% sequence overlap (`-c 0.9 -cov-mode 1 -min-seq-id 0.3`). All remaining sequences were clustered using MMseqs2 `cluster -c 0.9 -cov-mode 1 -min-seq-id 0.3` and appended to the database. We remove redundancy per cluster by keeping the most 10 diverse sequences using (`mmseqs filterresult --diff 10`). The final database consists of 209,335,865 million representative sequences and 738,695,580 members. See "Data availability" for input files. We extracted the MMseqs2 search workflow used in the server ("MSA generation by MMseqs2") into a standalone script `colabfold_search.sh` and provide it together with the databases.

Template information AlphaFold2 searches with HHsearch through a clustered version of the PDB (PDB70 [8]) to find the 20 top ranked templates. In order to save time, we use MMseqs2 [10] to search against the PDB70 cluster representatives as a prefiltering step to find candidate templates. This search is also done as part of the MMseqs2 API call on our server. Only the top 20 target templates according to E-value are then aligned by HHsearch. The accepted templates are given to AlphaFold2 as input features. This alignment step is done in the ColabFold client and therefore requires the subset of the PDB70 containing the respective HMMs. The PDB70 subset and the PDB mmCIF files are fetched from our server. For benchmarking, no templates are given to ColabFold.

Custom MSAs ColabFold allows researchers to upload their own MSAs. Any kind of alignment tool can be used to generate the MSA. The uploaded MSA can be provided in aligned FASTA, A3M, STOCKHOLM or Clustal format. We convert the respective MSA format into A3M format using the `reformat.pl` script from the HH-suite [8].

Modeling of protein-protein complexes Baek *et al.* [3] show that RoseTTAFold is able to model complexes, despite being trained only on single chains. This is done by providing a paired alignment and modifying the residue index. The residue index is used as an input to the models to compute positional embeddings. In AlphaFold2, we find the same to be true, although surprisingly the paired alignment is often not needed (**Fig. 2c**). AlphaFold2 uses relative positional encoding with a cap at $|i-j| \geq 32$. Meaning, any pair of residues separated by 32 or more are given the same relative positional encoding. By offsetting the residue index between two proteins to be >32, AlphaFold2 treats them as separate poly-peptide chains. ColabFold integrates this for modeling complexes.

For homo-oligomeric complexes (**Fig. 3a**), the MSA is copied multiple times for each component. Interestingly, it was found that providing a separate MSA copy (padding by gap characters to extend to other copies) to work significantly better than concatenating left-to-right.

For hetero-oligomeric complexes (**Fig. 3b**), a separate MSA is generated for each component. The MSA is paired according to the chosen `pair_mode` ("MSA pairing for complex prediction"). Since pLDDT is only useful for assessing local structure confidence, we use the fine-tuned model parameters to return the PAE for each prediction. As illustrated in **Supplementary Fig. 6**, the inter-PAE (predicted aligned error) or the predicted TM-score (derived from PAE) can be used to rank and assess the confidence of the predicted protein-protein interaction.

MSA pairing for complex prediction A paired MSA helps AlphaFold2 to predict complexes more accurately only if orthologous genes are paired with each other. We followed a similar strategy as Bryant *et al.* [21] to pair sequences according to their taxonomic identifier. For the pairing we search each distinct sequence of a complex against the UniRef100 using the same procedure as described in "MSA generation". We return only hits that cover all complex proteins within one species and pair only the best hit (smallest e-value) with an alignment that covers the query to at least 50%. The pairing is implemented in the new MMseqs2 module `pairaln`.

For prokaryotic protein prediction, we additionally implemented the protocol described in [3] to pair sequences based on their distances in the genome as predicted from the UniProt accession numbers.

Taxonomic labels for MSA pairing To pair MSAs for complex prediction, we retrieve for each found UniRef100 member sequence the taxonomic identifier from the NCBI taxonomy [27]. The taxonomic labels are extracted from the lowest common ancestor field ("common taxon ID") of each UniRef100 sequence from the `uniref100.xml` (2021_03) file.

Complex benchmark We compare predictions of five CASP14 complex targets (H1045, H1046, H1047, H1065, H1072) and 32 targets from Ovchinnikov *et al.* [22] to their native structures using MM-align [28] and extract TM-scores. We used `colabfold_batch` with BFD/MGnify and ColabFoldDB to predict structures in three different modes: (1) without MSA pairing, (2) with MSA pairing as described in "MSA pairing for complex prediction" and (3) with MSA pairing and also adding unpaired sequences. Models are ranked by pTMscore predicted by AlphaFold2.

Avoid recompiling AlphaFold2 models The AlphaFold2 models are compiled using JAX [29] to optimize the model for specific MSA or template input sizes. When no templates are provided, we compile once and, during inference, replace the weights from the other models, using the configuration of model 5. This saves 7 minutes of compile time. When templates are enabled, model 1 is compiled and weights from model 2 are used, model 3 is compiled and weights from models 4 and 5 are used. This saves 5 minutes of compile time. If the user changes the sequence or settings, without changing the length or number of sequences in the MSA, the compiled models are reused without triggering recompilation.

Avoid recompiling during batch computation In order to avoid AlphaFold2 model recompilation for every protein AlphaFold2 provides a function to add padding to the input MSA and templates called `make_fixed_size`. However, this is not exposed in AlphaFold2. We used the function in our batch notebook as well as in our command line tool *colabfold_batch*, in order to maximize GPU utilization and minimize the need of model recompilation. We sort the input queries by sequence length and process them in ascending order. We pad the input features by 10% (by default). All sequences that lie within the query length and an additional 10% margin do not require to be recompiled, resulting in a large speed up for short proteins.

Speed-up of predictions through early stop AlphaFold2 computes five models. We noted that for prediction of high certainty (>85 pLDDT), all five models would often produce structures of very similar confidence. In order to speed up the computation we added a parameter to `colabfold_batch` to define an early stop criterion that halts additional model inferences if a given pLDDT or pTMscore threshold is reached.

Recycle count AlphaFold2 improves the predicted protein structure by recycling (by default) 3 times, meaning the prediction is fed multiple times through the model. We exposed the recycle count as a customizable parameter as additional recycles can often improve a model at the cost of a longer runtime. We also implemented an option to specify a tolerance threshold to stop early. For some designed proteins without known homologous sequences, this helped to fold the final protein (**Supplementary Fig. 7**).

Sampling of diverse structures To reduce memory requirements, only a subset of the MSA is used as input to the model. Alphafold2, depending on model configuration, subsamples the MSA to a maximum of 512 cluster centers and 1024 "extra" sequences. Changing the random seed can result in different cluster centers and thus different structure predictions. ColabFold provides an option to iterate through a series of random seeds, resulting in structure diversity. Further structure diversity can be generated by using the original or fine-tuned (`use_ptm`) model parameters and/or enabling (`is_training`) to activate the stochastic (dropout) part of model. Enabling the latter, can be used to sample an ensemble of models for the uncertain parts of the structure prediction.

Proteome benchmark We predict the proteome of the archaeon *Methanocaldococcus jannaschii*. Of the 1787 proteins we exclude the 25 proteins longer than 1000 residues, leaving 1762 proteins of 268 aa average length. We search in 58 min using 100 threads on a system with 2x64-core AMD EPYC 7742 CPUs and 2TB RAM using `colabfold_search.sh` against the ColabFoldDB ("ColabFoldDB"), though we reduce the sensitivity to the considerably faster `-s 6` setting. We then predict the structures on a single Nvidia RTX 3090 with 28 GB RAM in 39.6 h using only MSAs (no templates). For each query we stop early if any model reaches a pLDDT of at least 85. We extrapolate the runtime for no-early-stopping by multiplying the runtime of model 3 for each protein to five models, yielding an overall speedup of factor 2.8. We observe a high structural agreement with an median TM-Score of 0.986 and mean TM-score of 0.953 when comparing the best predictions of ColabFold and AlphaFold2 with TMalign [30].

Benchmark with CASP14 targets We compare the AlphaFold-Colab and the AlphaFold2 (commit `b88f8da`) against ColabFold (commit `2b49880`, **Fig. 2**) using all CASP14 [2] targets. ColabFold uses UniRef30 (2021_03) [31] and the BFD/Mgnify or ColabFoldDB. AlphaFold-Colab uses the UniRef90 (2021_03), MGnify (2019_05) and the small BFD. AlphaFold2 uses the `full_dbs` preset with and default databases downloaded with the `download_all_data.sh` script. The 69 targets contain 96 domains, among these are 20 FM-targets with 28 domains. We compared the predictions against the experimental structures using TMalign [30].

Measuring time for CASP14 and complex targets All ColabFold and AlphaFold2 benchmarks were executed on systems with 2x16 core Intel Gold 6242 CPUs with 192 GB RAM and 4x Nvidia Quadro RTX5000 GPUs. Only one GPU was used in each individual run.

ColabFold was executed using `colabfold_batch`. The MMseqs2 server which computes MSAs for ColabFold has 2x14 core Intel E5-2680v4 CPUs and 768 GB RAM. Each generated MSA was processed by a single CPU-core. Runtimes were computed from server logs.

Runtimes for AlphaFold2 were extracted from the `features` entry of generated `timings.json` file. Where indicated with multicore, AlphaFold2 was used with the default 8 CPU cores for HMMer and 4 CPU cores for HHblits to process one query. For a fair comparison, AlphaFold2 was modified to allow HMMer and HHblits to access one CPU core.

AlphaFold-Colab was executed in the browser using a Google Colab Pro account. Times for homology search were taken from the log output of the "Search against genetic databases" cell in the notebook. The JackHMMer search uses 8 threads.

DATA AVAILABILITY

ColabFold databases are available at
`colabfold.mmseqs.com`.
Input databases used for building ColabFold databases:
UniRef30: `uniclust.mmseqs.com`
BFD: `bfd.mmseqs.com`
MGnify: `ftp.ebi.ac.uk/pub/databases/metagenomics/peptide_database/2019_05`
PDB70: `wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs`
MetaEuk: `wwwuser.gwdg.de/~compbiol/metaeuk/2019_11/MetaEuk_preds_Tara_vs_euk_profiles_uniqs.fas.gz`
SMAG: `www.genoscope.cns.fr/tara/localdata/data/SMAGs-v1/SMAGs_v1_concat.faa.tar.gz`
TOPAZ: `osf.io/gm564`
MGV: `portal.nersc.gov/MGV/MGV_v1.0_2021_07_08/mgv_proteins.faa`
GPD: `ftp.ebi.ac.uk/pub/databases/metagenomics/genome_sets/gut_phage_database/GPD_proteome.faa`

REFERENCES

[23] Kluyver, T. *et al.* Jupyter notebooks - a publishing format for reproducible computsteinegger2018ational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87–90 (IOS Press, 2016).

[24] Eastman, P. *et al.* *PLoS Comput. Biol.* **13**, 1–17 (2017).

[25] Krivov, G. G. *et al.* *Proteins* **77**, 778795 (2009).

[26] Suzek, B. E. *et al.* *Bioinformatics* **31**, 926–932 (2015).

[27] Federhen, S. *Nucleic Acids Res.* **40**, D136–D143 (2012).

[28] Mukherjee, S. & Zhang, Y. *Nucleic Acids Res.* **37**, e83–e83 (2009).

[29] Bradbury, J. *et al.* JAX: composable transformations of Python+NumPy programs (2018).

[30] Zhang, Y. & Skolnick, J. *Nucleic Acids Res.* **33**, 2302–2309 (2005).

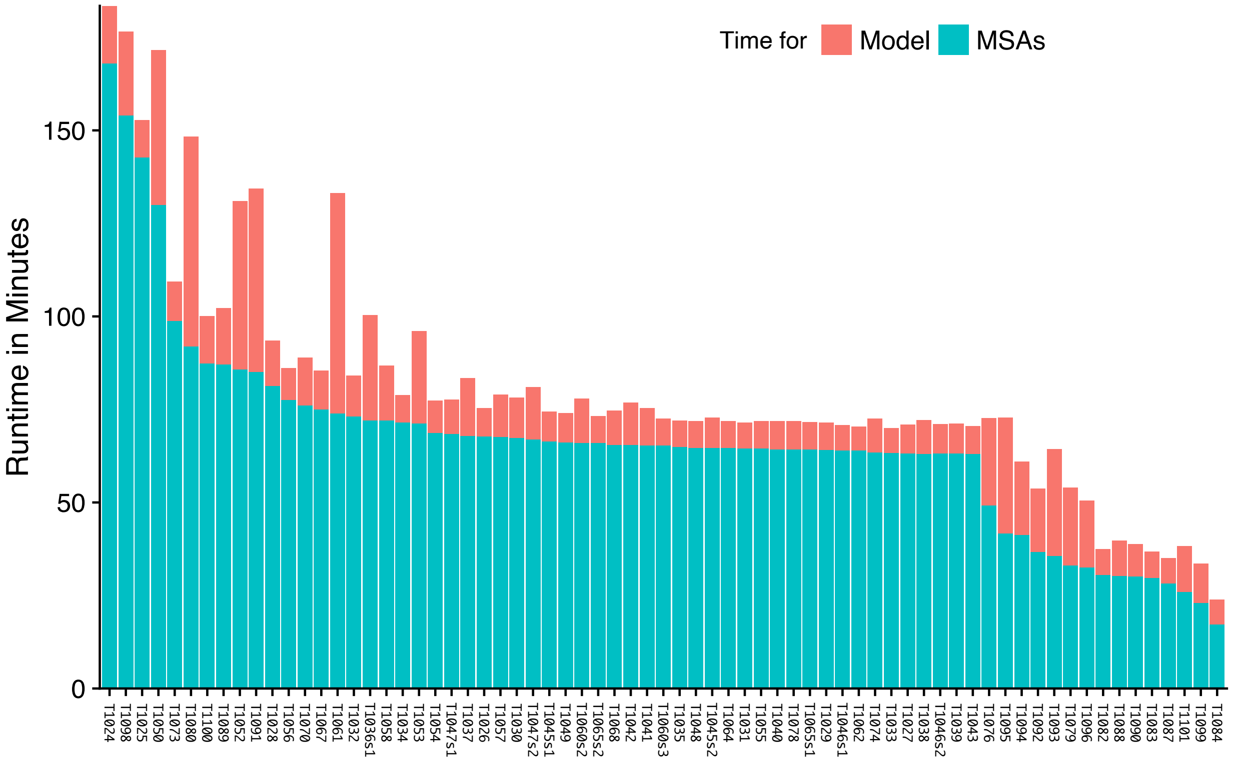
[31] Mirdita, M. *et al.* *Nucleic Acids Res.* **45**, D170–D176 (2017).

ColabFold - Making protein folding accessible to all

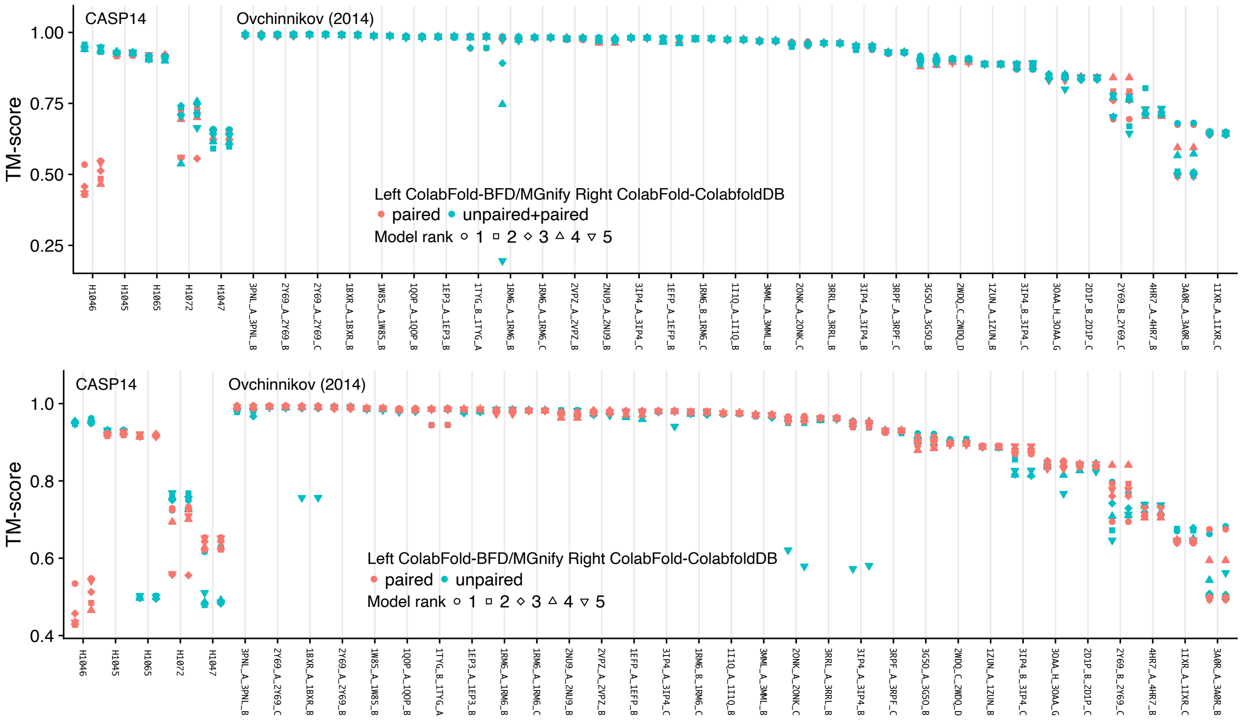
Milot Mirdita[1,*], Konstantin Schütze[2], Yoshitaka Moriwaki[3,4], Lim Heo[5], Sergey Ovchinnikov[6,7,*] and Martin Steinegger[2,8,*]

[1] Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany. [2] School of Biological Sciences, Seoul National University, Seoul, South Korea. [3] Department of Biotechnology, Graduate School of Agricultural and Life Sciences, The University of Tokyo, Tokyo, Japan. [4] Collaborative Research Institute for Innovative Microbiology, The University of Tokyo, Tokyo, Japan. [5] Department of Biochemistry and Molecular Biology, Michigan State University, East Lansing, MI 48824, USA. [6] JHDSF Program, Harvard University, Cambridge, MA 02138, USA. [7] FAS Division of Science, Harvard University, Cambridge, MA 02138, USA. [8] Artificial Intelligence Institute, Seoul National University, Seoul, South Korea [*] These authors contributed equally and are ordered alphabetically.

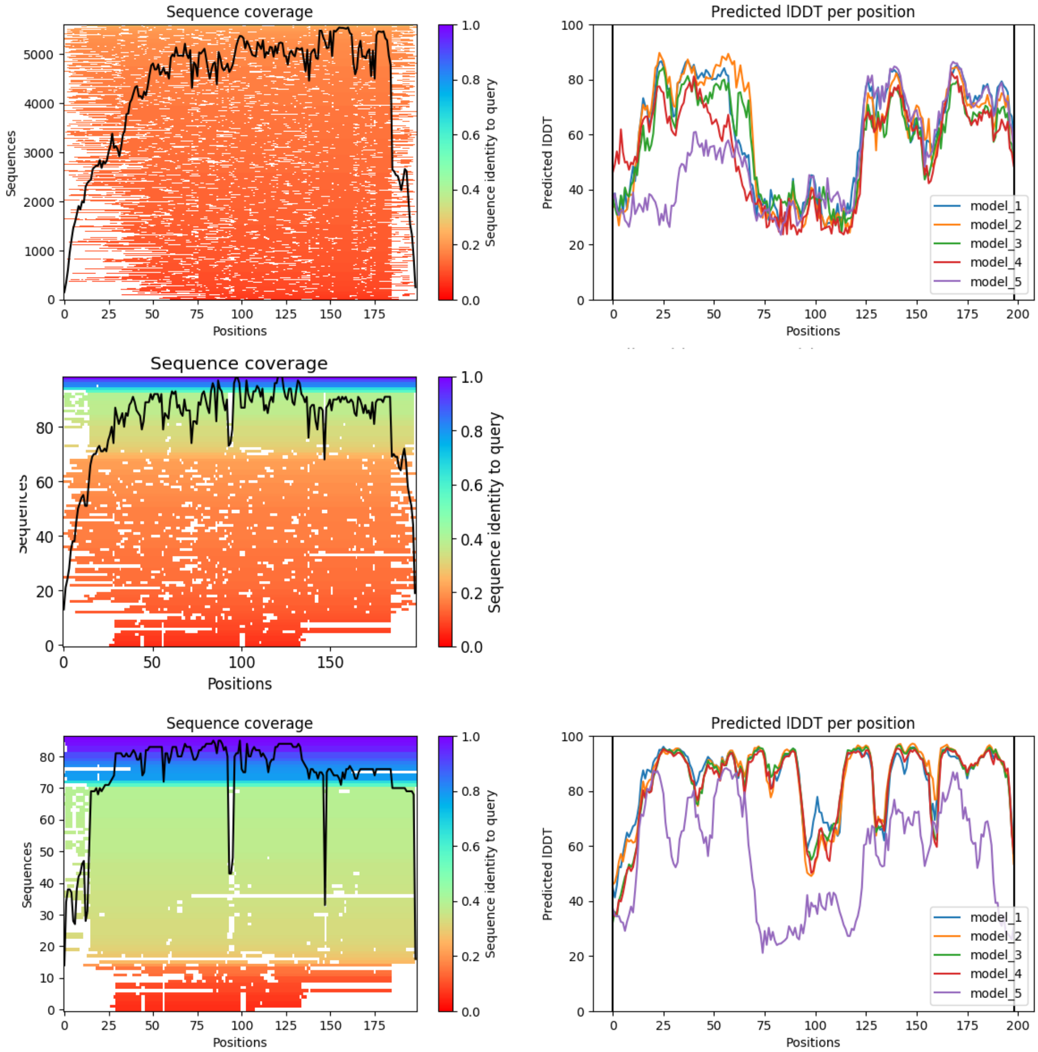
Contact: milot.mirdita@mpibpc.mpg.de, so@fas.harvard.edu, martin.steinegger@snu.ac.kr



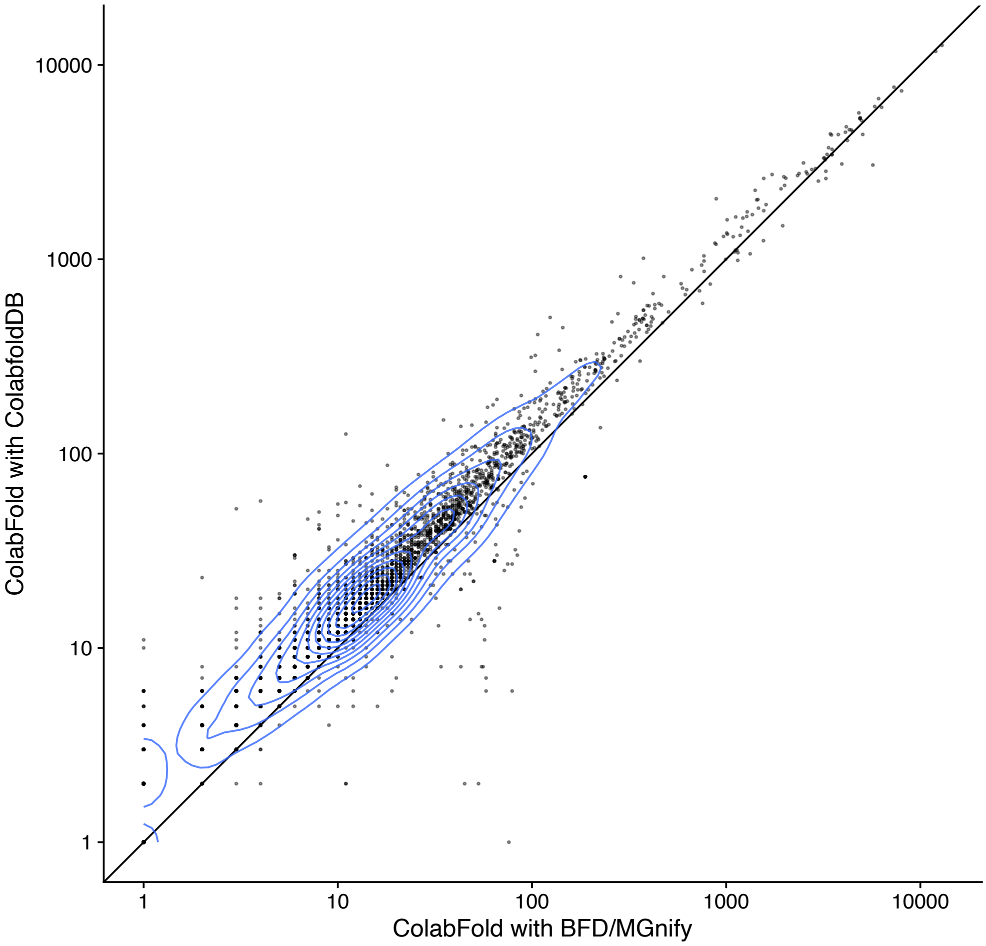
Supplementary Figure 1. **Time for inference vs MSA generation in AlphaFold2 pipeline (multi-core MSA generation)** 69 CASP14 targets were predicted using AlphaFold2 settings. Executed on systems with 2x16-core Intel Xeon 6242 with 192GB RAM and 4x NVIDIA RTX5000 (only one used for model inference).



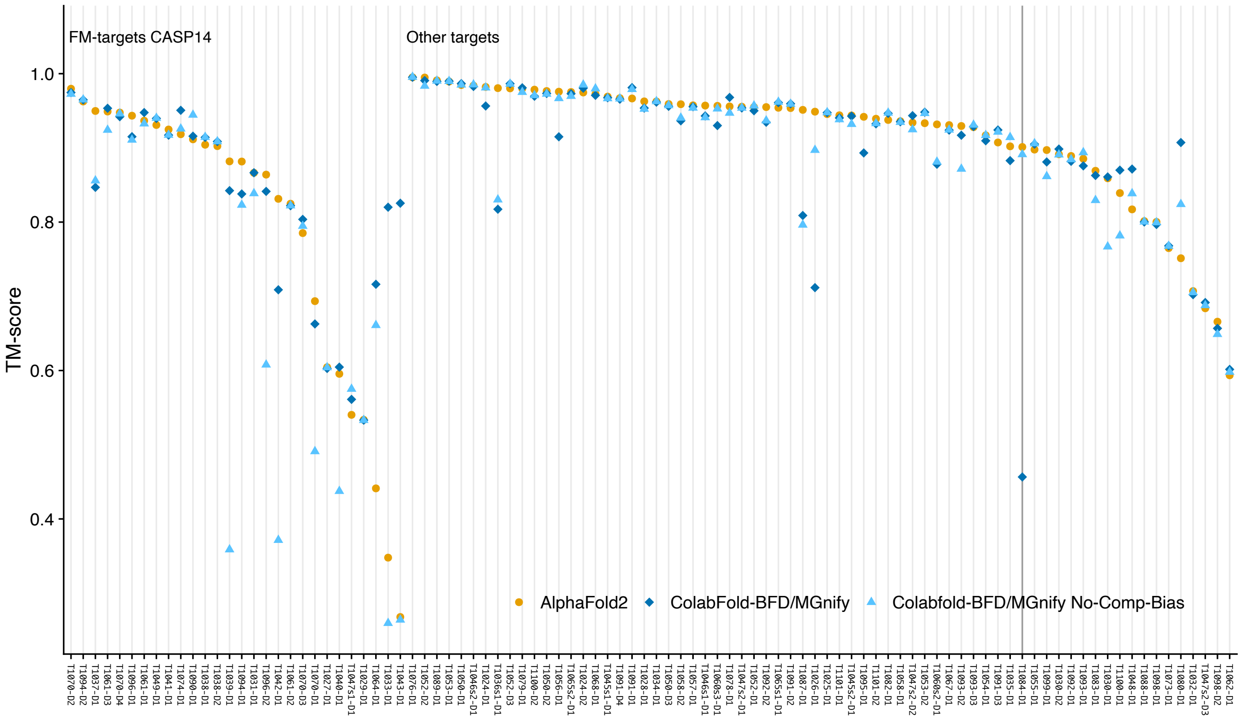
Supplementary Figure 2. Paired MSA complex prediction accuracy vs. unpaired and unpaired+paired.



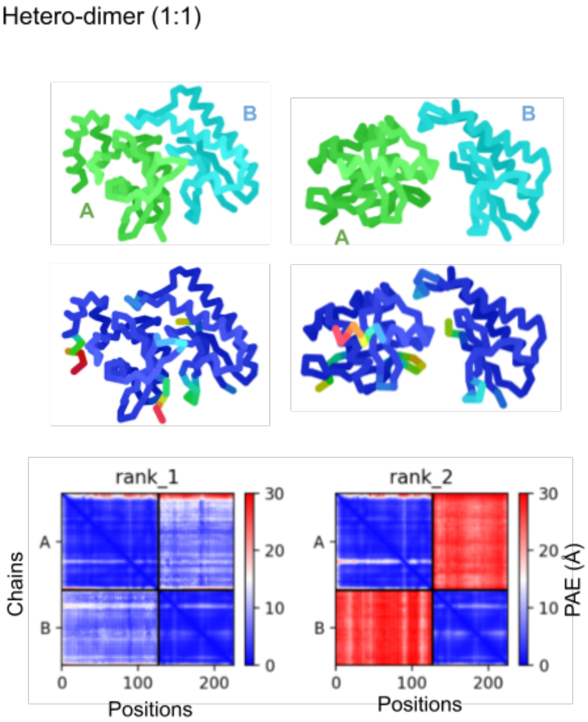
Supplementary Figure 3. **Anecdotal example of improved prediction through MSA filtering.** MSA coverage (left) and pLDDTs of predicted ColabFold models (right) for CASP14 target T1038 with two different filtering settings: Top: Single MSA filtering step with HHblits filtering algorithm and `--diff 3000` setting. Middle: Zoomed in view of first 100 sequences in top. Bottom: Three step MSA filtering as described in methods.



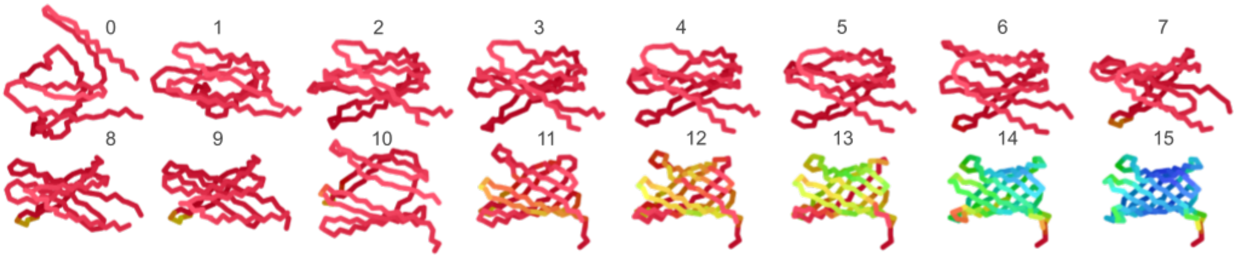
Supplementary Figure 4. **Comparison of Enrichment in PFAM** Comparison of homology search hits found from selected PFAM sequences against BFD/MGnify and ColabFoldDB. We select 2439 PFAM 34.0 entries that have less than 30 sequences in their `Pfam-A.full` entry. In each of these PFAM families we select from the `Pfam-A.seed` the longest sequence. We search this sequence with the ColabFold MMseqs2 workflow against the BFD/MGnify and ColabFoldDB. From the MSAs we cut the PFAM domains and note how many sequences cover the domain by at least 75%.



Supplementary Figure 5. **Disabling composition-bias and masking in MMseqs2 result in better accuracy for some CASP14 targets** We turned off two MMseqs2 mechanisms for false positive suppression (`--comp-bias-corr 0 --mask-profile 0`) and reran our CASP14 benchmark. Target T1084-D1 (highlighted) achieves now a TM-score of 0.891210 instead of 0.456540 in default search mode.



Supplementary Figure 6. **Hetero-dimer 1:1** Only one of the five models predicted for CASP14 target H1065 has a high agreement with its native structure during unpaired complex prediction. Although the pLDDT scores are nearly identical (shown in the middle with colored chains), the inter-PAE (bottom) is significantly lower (meaning more confident) for the correctly predicted complex (rank 1 vs rank 2). This demonstrates the utility of PAE (and the derived pTMscore) in ranking complexes.



Supplementary Figure 7. **Example of additional recycle steps improving prediction** Occasionally, increasing the number of recycles can help find a well predicted structure. For this de-novo designed transmembrane protein (Vorobieva et al. Science, 371(6531), 2021), 15 recycle iterations were needed to produce structure with high pLDDT.

5 Further contributions

5.1 Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold

Publication:

> Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold
>
> M. Steinegger[†], **M. Mirdita**, J. Söding[†]
>
> (†) corresponding author
>
> *Nature Methods* (2019), 16, 603–606.
> Cited 86 times since 07/2019.

Code and software availability

Plass is available as free open source software (GPLv3) at `plass.mmseqs.com`.

Author contributions

M.S. & J.S. designed research. M.S. & **M.M.** developed code and performed analyses. M.S., **M.M.** & J.S. wrote the manuscript.

Note

Due to licensing issues of the final manuscript, only the last version of the preprint is included here.

Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold

Martin Steinegger,[1, 2, 3, 4] Milot Mirdita,[1] and Johannes Söding[1, 4]

[1] *Quantitative and Computational Biology Group, Max-Planck Institute for Biophysical Chemistry, Am Faßberg 11, 37077 Göttingen, Germany*
[2] *Department of Chemistry, Seoul National University, Seoul, Korea*
[3] *Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, MD, USA*
[4] *Emails: martin.steinegger@mpibpc.mpg.de; soeding@mpibpc.mpg.de*

The open-source *de-novo* Protein-level assembler Plass (https://plass.mmseqs.org) assembles six-frame-translated sequencing reads into protein sequences. It recovers 2 to 10 times more protein sequences from complex metagenomes and can assemble huge datasets. We assembled two redundancy-filtered reference protein catalogs, 2 billion sequences from 640 soil samples (SRC) and 292 million sequences from 775 marine eukaryotic metatranscriptomes (MERC), the largest free collections of protein sequences.

A major limitation of metagenomic studies is that often a large fraction of short reads ($80\% - 90\%$ in soil [1]) cannot be assembled into contiguous sequences (contigs) long enough to allow for the prediction of gene and protein sequences. Because low-abundance genomes are difficult to assemble, the unassembled reads contain a disproportionately large part of the genetic diversity and probably an even greater share of biological novelty, which is mostly lost for subsequent analyses.

To decrease this loss and be less dependent on reference genomes, gene-centric approaches have been developed. Assemblies of hundreds of samples from one environment are pooled, genes in the contigs are predicted and clustered at $\sim 95\%$ identity into gene catalogs [2–4]. Gene abundances in each sample are found by mapping reads to the reference gene clusters. In this way, the functional and taxonomic composition of metagenomic samples and their dependence on environmental parameters can be studied. Also, genome-based analyses are enabled by abundance binning, which finds sets of catalog genes with correlated abundances across many samples and hence likely to belong to the same genome [5, 6].

State-of-the-art assemblers for metagenomic short reads [7–9] find contigs as paths through a de-Bruijn graph. This graph has a node for each k-mer word in the reads and edges between k-mers occurring consecutively in a read. On metagenomic data, de-Bruijn assemblers suffer from a limited sensitivity-selectivity trade-off: k-mers have to be long and specific to avoid the graph exploding with false edges. But long k-mers lack sensitivity when intra-population diversities are high and overlapping reads often contain mismatches due to single nucleotide polymorphisms (SNPs). Whatever k is chosen, k-mers will be too short to be specific enough in genomic regions conserved between species and too long to be sensitive enough in regions of high intra-population diversity. This dilemma leads to short, fragmented assemblies.

Most SNPs in microbial populations lead to no change or conservative substitutions in the encoded protein sequences (**Supplementary Fig. 1**). ORFome [10] and SFA-SPA [11] therefore proposed to assemble protein instead of nucleotide sequences. But they are too slow to run on large metagenomes and as de-Bruijn assemblers they suffer from the same limited specificity-sensitivity trade-off.

In addition to avoiding mismatches, assembling protein sequences also circumvents the major issue in genome assembly, sequence repeats, because proteins have much fewer and shorter repeats. Furthermore, chimerical assemblies between similar protein sequences (say $\geq 97\%$ sequence identity) are much less problematic in that they do not lead to false conclusions about which genes occur together in a genome. Therefore, protein-level assembly also increases coverage by assembling sequences that cannot be assembled on the nucleotide level due to the risk of chimeric assemblies.

Plass uses a novel graph-free, greedy iterative assembly strategy (**Fig. 1**) that, together with its linear-time all-versus-all overlap computation (steps 2-4) [12], scales linearly in runtime and memory. This permits the overlap-based assembly of huge read sets on a single server. Most importantly, by computing full alignment overlaps instead of only k-mer matches, Plass overcomes the specificity-sensitivity limitation of de-Bruijn assemblers, allowing it to recover several times more proteins sequences from complex metagenomes.

Plass needs to keep the protein sequences in main memory to avoid random disk access (step 4). It therefore needs 1 byte of memory for every amino acid translated from the input reads, or $\sim$500 GB RAM to assemble 2-3 billion 2×150 bp reads. In comparison, memory requirements and runtimes of overlap graph assemblers scale superlinearly with the number of reads. Plass therefore combines the high specificity-sensitivity of overlap graph assemblers with the linear runtime and memory scaling of de-Bruijn graph assemblers.

Due to our greedy assembly approach, the most critical aspect to analyze is what fraction of the sequences are wrongly assembled (precision). To challenge Plass, we sought two hard datasets containing many related genomes, as this increases the risk of chimeric assemblies [13].

The first set consists of 96 single-cell assembled genomes of *Prochlorococcus* [14] taken from a single sample of seawater.

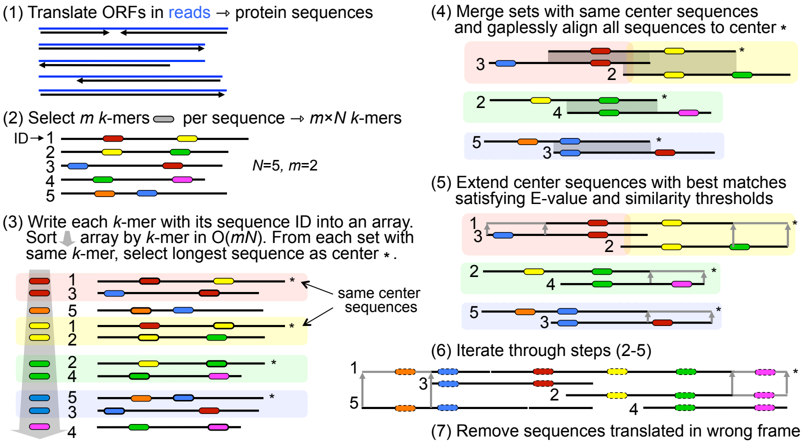

(1) Translate ORFs in reads → protein sequences
(2) Select m k-mers per sequence → m×N k-mers
ID→1
2
3
4
5
N=5, m=2
(3) Write each k-mer with its sequence ID into an array. Sort array by k-mer in O(mN). From each set with same k-mer, select longest sequence as center *.
same center sequences
(4) Merge sets with same center sequences and gaplessly align all sequences to center *
(5) Extend center sequences with best matches satisfying E-value and similarity thresholds
(6) Iterate through steps (2-5)
(7) Remove sequences translated in wrong frame

FIG. 1. **Plass workflow.** (1) Merge overlapping read pairs and translate all potential ORFs with $\geq$45 codons into protein sequences. (2) For each of these N sequences, select the m k-mers with the lowest hash values (default: $m = 60$, $k = 14$, reduced alphabet size $= 13$). Write the mN k-mers into an array together with sequence identifiers. (3) Sort the array by k-mer to find for each k-mer the set of sequences containing it, and assign the longest sequence as the set's center. (4) Resort the array by center sequence into groups and gaplessly align the center sequence to each group member ($< mN$ alignments). Remove sequences with insufficient E-value (default: $> 10^{-5}$) from the group. (5) Iteratively extend each center sequence by the remaining group member with highest sequence identity (default: $\geq$ 90%) until all group members have been processed. (6) Iterate steps 2 to 5 (default: 12 times). (7) Remove sequences translated in the wrong frame using a neural network.

These cyanobacteria are known for their high intra-species genetic diversity. The second, very hard set contains 738 single-cell assembled genomes, 489 of which are *Prochlorococcus*, 50 are *Synechococcus*, and 199 are genomes from a diverse range of prokaryotic and viral groups [15].

As ground truth reference we predicted protein sequences on the genomes using Prodigal [16]. We simulated 2 × 150 bp reads with a mean coverage of 1 for each genome.

We assembled protein sequences from these nucleotide reads using Plass and SFA-SPA [11]. We assembled nucleotide contigs with three widely used nucleotide assemblers, Megahit [8], metaSPAdes [9], and Velvet [7], the first two of which were among the top assemblers in recent benchmarks [13, 17, 18]. We predicted protein sequences in their contigs using Prodigal and, to ignore unassembled reads, we removed protein sequences with less than 100 residues.

The assembly sensitivity is the fraction of amino acids in the reference proteins that have a sequence match with at least $X\%$ sequence identity with an assembled sequence. To avoid giving too much weight to highly conserved proteins, we redundancy-filtered the reference proteins for the sensitivity analysis using Linclust [12] with 95% sequence identity. The sensitivity is similar for the three nucleotide assemblers, whereas Plass assembles up to 56% more residues correctly than the next best tool, metaSPAdes (**Fig. 2a, top**).

The Plass-assembled proteins cover over 80% of the Megahit and metaSPAdes assemblies at 99% minimum sequence identity, whereas the latter cover only around 40% of the Plass assembly at this cut-off (**Supplementary Fig. 3**).

The precision is the fraction of assembled amino acids that have at least $X\%$ sequence identity with an assembled sequence.

Since ORFs are predicted on the nucleotide assemblies using the same tool used to define the reference protein sequences on the single-cell genomes, whereas Plass uses a very different approach (Online Methods), the benchmark is biased against Plass. Nevertheless, Plass achieves the same precision below $X\% = 97\%$, where all assemblers except SFA-SPA achieve similar precision (**Fig. 2a, middle**). The 2% − 7% of missing precision at $X\% = 90\%$ are mainly caused by mispredicting open reading frames (ORFs) in the assembled sequences or on the single-cell genomes.

Plass' neural network filter (Online Methods) for suppressing proteins translated in wrong frames raises the precision at $X\% = 90\%$ on the very hard set by a few percent (**Supplementary Fig. 3b**).

Plass filters out sequences translated from wrong ORFs based on their amino acid and dipeptide composition, which differs from correctly translated, real protein sequences. We-trained a neural network using as features the20length-normalized amino acid frequencies of the sequence and the62length-normalized dipeptide frequencies in a reduced alphabet of size6. Our fully connected network has56input nodes,a hidden layer of96nodes, and a single output node.

However, Plass produces much fewer proteins at 99% sequence identity than the nucleotide assemblers, particularly on the very hard data set (**Fig. 2b**). Increasing the sequence identity threshold for merging sequence fragments from 90% to 97% (pink trace) markedly improves sensitivity and precision on both datasets, but still much fewer proteins match a reference protein with identity of $\geq$ 97%.

To test the impact of the assembly of chimeric protein sequences on the quality of functional annotations, we annotated each protein sequence assembled from the simulated reads and each reference protein from the single-cell genomes with an orthologous group using the eggNOG mapper [19]. We compared the eggNOG annotation of each assembled protein sequence with the annotation of the best-matching sequence found in the reference protein set and scored a true positive (TP) if annotations matched and a false positive (FP) otherwise. Assemblies that can not be assigned to a reference are FPs. Despite Plass' lower assembly precision, annotations of its proteins achieve lower false discovery rates (FP/(TP + FP)) than those of the other assemblers, on both data sets (**Fig. 2, bottom**). We believe this is due to (1) the high conservation of

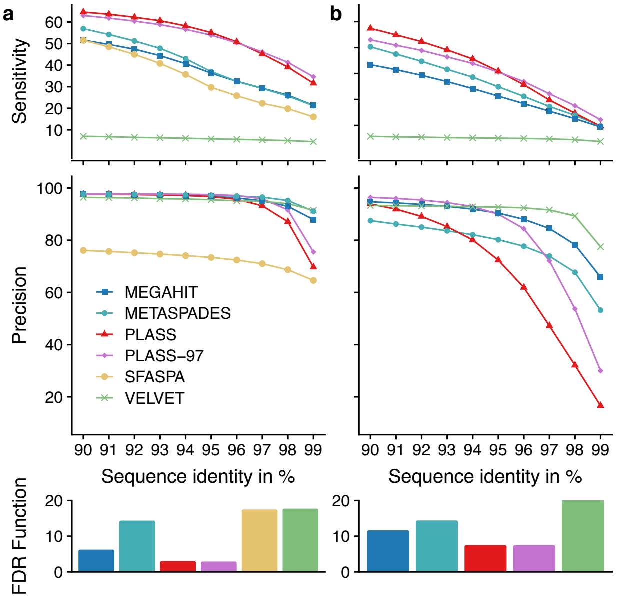


FIG. 2. **Plass assembles many times more protein sequences from various environments than the state of the art.** (**a**) Sensitivity and precision of protein sequences assembled from synthetic reads sampled (**a**) from 96 assembled genomes of single *Prochlorococcus* cells [14] and (**b**) from 738 single-cell assembled genomes of diverse marine prokaryotes and viruses [15]. For the three nucleotide assemblers we predicted protein sequences on their assembled contigs with Prodigal [16]. Top: Sensitivity is the fraction of reference sequence amino acids that have are aligned to an assembled protein sequence with a sequence identity at least the value on the x-axis. Middle: Precision is the fraction of assembled amino acids that are aligned to a reference protein with sequence identity at least the value on the x-axis. Bottom: False discovery rate (1 − precision) of orthology-based functional annotations of assembled proteins. Colors and order of tools as in previous legend.

molecular and cellular functions at sequence identities $> 90\%$ (above 60% identity, 90% of proteins conserve of all four EC number digits [20]), (2) the limited ability of homology-based function annotation tools to predict the effects of point mutations, and (3) the positive impact of more complete protein sequences on the prediction accuracy.

On real metagenomic datasets, no ground-truth set of reference sequences exists. Therefore precision can not be measured, but sensitivity in terms of the total number of assembled amino acids can be compared. We used four representative test sets: a single 11.3 Gbp sample from the human gut [21], 775 samples with 15 Tbp of eukaryotic metatranscriptome reads from TARA [22], a 31 Gbp sample from Hopland grass soil (*Brodie et al.*, unpublished), and 538 Gbp of reads in 12 samples from the same project to test the benefits of co-assembly (**Fig. 2b-e**). All datasets contain 2×150 bp overlapping paired-end sequences, except the metatranscriptomics sample, which has 2×102 bp reads.

We compared the Marine Eukaryotic Reference Catalog (MERC) assembled by Plass to the Marine Atlas of Tara Oceans Unigenes (MATOU) [22] assembled by Velvet. The gut and soil datasets could not be assembled with Velvet due to insufficient memory, and we could only compare Plass to Megahit and metaSPAdes. The twelve Hopland soil datasets with 1.5 billion reads could be co-assembled in one go by Plass. Megahit raised an out-of-memory exception, therefore we assembled each sample separately and pooled the contigs. For human gut, Hopland soil, and the soil co-assembly, Plass took 4h 20min, 6h 20min and 360h respectively, while Megahit took 3h, 21h 30min and 200h respectively.

On the gut sample, Plass assembled 32% more amino acids than Megahit/Prodigal at a length cut-off of 100 amino acids (**Fig. a-d**, top). The marine eukaryotic reference catalog (MERC) assembled by Plass is 2.8-fold larger than MATOU assembled by Velvet, and on the Hopland soil data Plass assembled 2.7 times more than Megahit. In the soil co-assembly, Plass co-assembled 10 times more amino acids than the pooled assembly of Megahit. The increase of the ratio with sequence length in the top of **Fig. 2d,e** indicates that the sequences assembled by Plass are significantly longer than those of Megahit/Prodigal. These gains in recovered protein sequences are similar at all levels of redundancy up to 80% sequence identity (bottom half of **Fig. 2b-e**).

We wanted to know how strongly the improved sensitivity of Plass affects the apparent taxonomic composition. We implemented the 2bLCA protocol (**Supplementary Fig. 4**) [23] to map each read via its translated ORF to an assembled protein sequence and each protein sequence to a node in the taxonomic tree. By transitivity this maps reads to taxonomic nodes. The absolute number of reads mapped to various taxonomic nodes (**Supplementary Fig. 5a**) is around twice higher for Plass than for Megahit/Prodigal. Remarkably, the distribution over taxa can deviate quite substantially between the assembly methods (**Supplementary Fig. 5b**), which could be caused be a systematic dependence of the assembly sensitivity on genomic coverage (**Supplementary Fig. 5c**).

Plass is well suited to large-scale applications. We assembled a Soil Reference protein Catalog (SRC) from 18 Tbp of reads from all 640 soil samples that were sequenced between 01/2016 and 02/2018 using Illumina HiSeq or NovaSeq with 2×150 bp paired-end reads. Each sample was assembled on a server with 2×8 cores and 128 GB memory, resulting in 12 billion protein sequences after a total runtime of about six weeks on 25 servers. We clustered the sequences to 90% sequence identity at 90% minimum coverage using Linclust [12], resulting in 2 billion sequences with an average length of 163 amino acid residues. Among those, at least 52.3 million sequences are complete, meaning that Plass found the stop codon and the earliest possible start codon (Online Methods). This dataset contains 6.8, 4.0 and 3.9 times more amino acids than the Uniprot database after redundancy-filtering both databases at 90%, 70% and 50%, respectively (**Fig. 3a**).

To assess the degree to which the SRC represents the diversity of soil metagenomes, we selected two soil samples not used for building the SRC, randomly sampled and merged 10 000 overlapping 2×150 bp read pairs, predicted protein sequences with Prodigal, and searched with these through the

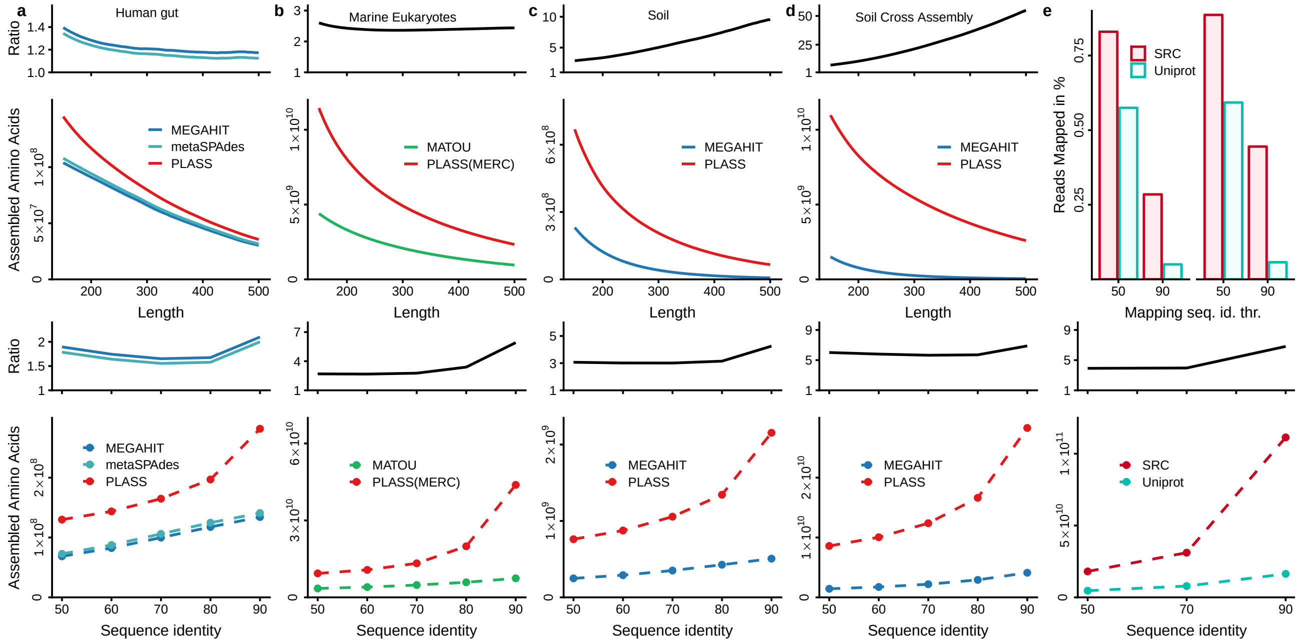


FIG. 3. **Plass assembles many times more protein sequences from various environments than the state of the art. (a-d)** Total number of amino acids in redundancy-filtered sets of protein sequences assembled by Plass (red traces) compared to the total number of amino acids of redundancy-filtered protein sequences predicted by Prodigal on contigs assembled by Megahit (**a,c,d**: blue) or on contigs in the eukaryotic metatranscriptomes reference assembly (**b**: green) [22]. Top half: dependence on the minimum protein sequence length using a redundancy-filtering with 80% maximum pairwise sequence identity. Bottom half: dependence on the strength of redundancy filtering for a minimum sequence length of 100 amino acids. Black traces: fold increase in total assembly length by Plass versus the state of the art. (**e**) Top half: Fraction of reads sampled from two soil metagenomes (not included in the SRC) that could be mapped with 90% and 50% minimum sequence identity to a sequence in the SRC or in Uniprot. Bottom Half: Numbers of amino acids in SRC (red) and Uniprot (turquoise) and their ratio (black) after redundancy-filtering with a maximum pairwise sequence identity of 50%, 70% and 90%.

90%-redundancy-filtered versions of the SRC and the Uniprot, using the `map` workflow of MMseqs2 [24]. **Fig. 3b,c** shows the fraction of reads that obtained matches with at least 50% and 90% sequence identity. At 50% threshold, 82.5% and 89.5% of the soil reads matched to the SRC in the two samples, while only 62% and 64% matched to the Uniprot.

The chief limitation of Plass is that, unlike nucleotide assemblers, it cannot place the assembled protein sequences into genomic context. Furthermore, Plass relies on six-frame translation. It therefore cannot assemble intron-containing eukaryotic proteins from metagenome data, although, as shown, it can assemble eukaryotic proteins from transcriptome data. Another important drawback is its inability to resolve homologous proteins with sequence identities above ∼95%, for example originating from closely related strains or species. However, in our tests this had little impact on the accuracy of predicted functions (**Fig. 2**). Also, one can argue that bacterial phenotypes are determined mainly by the complement of horizontally acquired accessory genes such as virulence factors, much more so than by minor variations in protein sequences. Whereas Plass is clearly worse than nucleotide assemblers in resolving such variation, it excels at assembling more complete protein complements of metagenomes.

In conclusion, Plass is well-suited for very large-scale metagenomic applications, for example to generate reference protein sequence catalogs for every major type of environment. In addition to facilitating metagenomics analyses, these catalogs can be mined for proteins of interest to biotechnology or pharmacology. They will also improve homology detection, protein function annotation and protein structure prediction [25] by enriching multiple sequence alignments with diverse homologs.

Methods

Methods, including statements of data availability and references, are available in the online version of the paper.

Acknowledgements

We are grateful to Cedric Notredame and Chaok Seok for hosting MS at the CRG and SNU for 12 and 30 months, respectively. We thank Shinichi Sunagawa, Folker Meyer and Alex Sczyrba for helpful discussions and Titus Brown for his early analysis and detailed feedback on Plass results. This work was supported by the EU's Horizon 2020 Framework Programme (Virus-X, grant 685778).

Author contributions

MS & JS designed research, MS & MM developed code and performed analyses, MS, MM & JS wrote the manuscript.

[1] Howe, A. C. *et al.* *Proc. Natl. Acad. Sci. U.S.A.* **111**, 4904–4909 (2014).

[2] Li, J. *et al.* *Nat. Biotechnol.* **32**, 834–841 (2014).

[3] Sunagawa, S. *et al.* *Science* **348** (2015).

[4] Xiao, L. *et al.* *Nat. Biotechnol.* **33**, 1103–1108 (2015).

[5] Nielsen, H. B. *et al.* *Nat. Biotechnol.* **32**, 822–828 (2014).

[6] Forslund, K. *et al.* *Nature* **528**, 262 (2015).

[7] Zerbino, D. & Birney, E. *Genome Res.* **18**, 821–829 (2008).

[8] Li, D. *et al.* *Bioinformatics* **31**, 1674–1676 (2015).

[9] Nurk, S. *et al.* *Genome Res.* **27**, 824–834 (2017).

[10] Ye, Y. & Tang, H. *J. Bioinform. Comput. Biol.* **7**, 455–471 (2009).

[11] Yang, Y. *et al.* *Bioinformatics* **31**, 1833–1835 (2015).

[12] Steinegger, M. & Söding, J. *Nat. Commun.* **9**, 2542 (2018).

[13] Sczyrba, A. *et al.* *Nat. Methods* **14**, 1063–1071 (2017).

[14] Kashtan, N. *et al.* *Science* **344**, 416–420 (2014).

[15] Berube, P. M. *et al.* *Scientific Data* **5**, 180154 (2018).

[16] Hyatt, D. *et al.* *BMC Bioinformatics* **11**, 119 (2010).

[17] Vollmers, J. *et al.* *PloS one* **12**, e0169662 (2017).

[18] van der Walt, A. J. *et al.* *BMC Genomics* **18**, 521 (2017).

[19] Huerta-Cepas, J. *et al.* *Mol Biol Evol* **34**, 2115–2122 (2017).

[20] Tian, W. & Skolnick, J. *Journal of molecular biology* **333**, 863–882 (2003).

[21] Lee, S. T. M. *et al.* *Microbiome* **5**, 50 (2017).

[22] Carradec, Q. *et al.* *Nat. Commun.* **9**, 373 (2018).

[23] Hingamp, P. *et al.* *ISME J.* **7**, 1678–1695 (2013).

[24] Steinegger, M. & Söding, J. *Nat. Biotechnol.* **35**, 1026–1028 (2017).

[25] Ovchinnikov, S. *et al.* *Science* **355**, 294–298 (2017).

ONLINE METHODS

Plass proceeds in seven steps summarized in **Fig. 1**.

Merging paired-end reads and ORF calling. Longer reads increase the precision and sensitivity of the assembly due to longer overlaps obtaining higher statistical significance. In step 1, Plass therefore merges overlapping paired-end reads into longer sequences using code from the open-source FLASH tool [23], which we integrated into Plass (step 1**a**).

Furthermore, in step 1, Plass extracts all open reading frames (ORFs) with at least 45 codons and translates them into protein sequences. (Alternative codon tables can be specified with option `-translation-table`.) To determine the correct start codon later on, it also extract and translates all ORFs with at least 20 codons starting with a putative `ATG` start codon that is the first `ATG` codon after a stop codon in the same frame. Because the coding sequence cannot start before such an `ATG`, these sequences help Plass to predict start codons later on (see "Predicting start codons" and **Supplementary Fig. 6**).

Finding overlaps in linear time. The identification of all overlapping alignments (**Fig. 1**, steps 2-4) is critical for the performance of overlap assemblers. Previously proposed protein-level assemblers have a runtime complexity that scales quadratically with the input set size [24, 25]. A typical metagenomic read set with 100 million reads requires 10^{16} comparisons with a quadratic method. To speed up the computation, we adapted our linear-time clustering algorithm Linclust [26] for assembly.

In step 2, Plass transforms each protein sequence into a reduced amino acid alphabet, whose 13 letters represent the following groups of amino acids: (L, M), (I, V), (K, R), (E, Q), (A, S, T), (N, D), and (F, Y). From each reduced sequence it selects m (default: $m = 60$) k-mers (or $l - k + 1$ if the sequence of length l contains only $l - k + 1 < m$ k-mers). The selected k-mers are those with lowest hash values. Our rolling hash function [26] maps each k-mer onto a range of $[0, 2^{16}]$ such that even single residue changes result in quasi-random, unrelated hash values. For each of the $\sim mN$ selected k-mers, Plass stores in an array the k-mer index (8 bytes), the sequence identifier (4 bytes), the k-mer position in the sequence (2 bytes) and the length of the sequence (2 bytes).

In step 3, Plass sorts the array by k-mer index and sequence length to find the sets of sequences containing the same k-mer. For each set, it picks the longest sequence as the center sequence. For each member of the k-mer set, it overwrites the k-mer index with the center sequence identifier and computes the diagonal $i - j$ on which the shared k-mer match occurs, where i is the k-mer position in the center sequence and j is the position in the member sequence. The array now contains the center sequence identifier, the member sequence identifier, the k-mer match diagonal, and the length of the member sequence. It sorts the array again, this time by center sequence identifiers, and removes duplicate center-member pairs. If more than one diagonal match between a center and member sequence is found only the match with lowest diagonal is kept.

In step 4, Plass computes an ungapped local alignment between each center sequence and each group member, using one-dimensional dynamic programming on the diagonal $i-j$ of the k-mer match. It computes E-values using ALP [27] and, by default, the Blosum62 substitution matrix. Alignments with an E-value $> 10^{-5}$ (default) and a sequence identity $< 90\%$ (default) are rejected.

Extending protein reads. In step 5, Plass extends the center sequence by concatenating the non-overlapping residues of the member sequence with highest similarity in the overlap. More precisely, it processes the list of alignments with the member sequences in order of descending overlap sequence identity, until one side of the center sequence has been extended and the other side has either been extended as well or has no extending alignments left in the list. Then it realigns the extended center sequence with all yet unprocessed member sequences and iterates the extension until the entire list of alignments has been processed.

Iterative assembly. Plass iterates through steps 2 to 5 twelve times (default), each time updating the original version of the center sequences with their extended versions and keeping all other sequences unchanged (step 6). To extract different k-mers in each new iteration, we increment the step size of the circular shift inside our rolling hash function [26].

Removing proteins translated in wrong frames. In step 7, Plass removes sequences translated from wrong ORFs or assembled from such sequences. ORFs translated in the wrong frame contain a stop-codon approximately every $64/3 \approx 21$ residues, and so only a fraction of around $\exp(-45/21.3) \approx 12\%$ contain ≥ 45 codons.

Plass filters out sequences translated from wrong ORFs based on their amino acid and dipeptide composition, which differs from correctly translated, real protein sequences. We trained a neural network using as features the 20 length-normalized amino acid frequencies of the sequence and the 6^2 length-normalized dipeptide frequencies in a reduced alphabet of size 6. Our fully connected network has 56 input nodes, a hidden layer of 96 nodes, and a single output node. We trained the network using the Keras deep-learning framework using the Adam optimizer with a 10% drop-out probability and the binary cross-entropy loss function. We leave 10% of the data out for cross validation. The network is integrated into Plass using Kerasify.

To train the network, we created a positive set of known coding sequences and a negative set of sequences translated in a wrong reading frame. The positive set contained 2.4 million proteins sampled from the prokaryotic subset of the Uniclust30 representative sequences [28]. For the negative set, we extract all ORFs from 757 prokaryotic genomes contained in the KEGG database [29] and clustered them using MMseqs2 [30] with a maximum sequence identity of 30% and a minimum coverage of 80%. Clusters without any member with coding sequence annotation in KEGG or homology to entries in Uniclust30 (requiring an E-value of $< 10^{-3}$) were extracted. From these, we sampled 2.4 million sequences.

Predicting start codons. To determine the correct start codon and minimize overextension at the N-terminus in the ORF translation step 1, Plass marks with a prepended asterisk $*$ those methionine residues that represent the first `ATG` after a stop codon in the same frame, as this implies that the coding sequence can at the earliest start there (**Supplementary Fig. 6**).

After the alignment and extension step 4 in the first iteration, Plass reconstructs the multiple sequence alignment of all merged sequences. Where at least 20% of all methionines in a column are marked by a prepended asterisk, it removes the preceding residues from all other sequences and prepends an asterisk to all sequences to mark the start. If several columns fulfill the 20% criterion, it trims the sequences at the most downstream of these columns. The start codon prediction is only done in the first iteration to save time and disk space.

Suppressing repetitive sequences. Protein repeats can lead to unwanted extensions during assembly. We therefore detect sequences with repeat regions during step 2 as those containing at least 8 (default) identical k-mers (in the 13-letter alphabet). These sequences are ignored during all steps.

Memory-efficient processing of huge input sets. Plass needs 1 byte per residue of translated protein sequence generated in step 1 to keep these sequences in memory and avoid random disk accesses in the alignment step 4. But as we saw, the k-mer array in steps 2 and 3 occupies $m \times 16\,\text{bytes} = 720\,\text{bytes}$ of memory per sequence, which is around 16 times more than 1 byte per residue. We removed this bottleneck, for systems with insufficient main memory, by splitting the k-mer array into a number of chunks and processing them sequentially, with little loss in speed.

We compute the maximum number of chunks S that still allows one chunk to fit into the available system memory M as $S = \lceil mN \times 16\,\text{byte}/M \rceil$. For each chunk c out of S, we proceed with steps 2 and 3 exactly as described before except that we only extract k-mers whose index R satisfies ($R \bmod S$) $= c$ and that we store the chunks of k-mer arrays on hard disk. After all splits have been computed, we merge them into a single k-mer array.

Assembly quality benchmark. We could not use the standard benchmark developed by the Critical Assessment of Metagenomic Interpretation (CAMI) [31], because the mutation model of the sgEvolver tool used for simulating population microdiversity (strain diversity) does not penalize frame-disrupting indels and non-conservative substitutions within coding regions. This leads to very low and unrealistic conservation of coding regions. The synthetic reads generated with such a model are certainly realistic enough to test nucleotide assemblers but would render protein-level assembly absurdly unsuitable.

We sought to construct a genomic benchmarking set that would contain a high degree of *natural* variation, in which the genomic sequences reflect the actual evolutionary pressures on them.

We downloaded from the sequence read archive (SRA) at the NCBI/NIH two sets of genomes assembled from single cell sequencing libraries. The first set contains genomes of 96 *Prochlorococcus* genomes [32]. These cells were taken from the same ocean water sample and represent a population of the cyanobacteria *Prochlorococcus*, the most abundant marine photosynthetic organism on earth noted for high intra-species diversities. Sequence identities of 16S rRNA ITS sequences in a matched sample are between 50% and 100%.

The second set contains 738 single-cell genome assemblies (NCBI project PRJNA445865) consisting of 489 *Prochlorococcus*, 50 *Synechococcus*, 82 SAR11, 17 SAR116, 16 SAR86, 9 extracellular virus particles, and 75 additional sympatric microorganisms, sampled at 22 locations in the Atlantic and Pacific Oceans [33].

As ground truth reference we predicted protein sequences on the genomes using Prodigal [34] and removed sequences shorter than 100 residues, resulting in a redundant reference set of 109 014 protein sequences for set 1 and 829 899 for set 2. We reduced the redundancy by clustering with Linclust at 95% sequence identity and 99% minimum coverage of the shorter sequence (options `--cov-mode 1 -c 0.99 --min-seq-id 0.95`), resulting in the non-redundant reference set with 14 943 (set 1) and 460 653 (set 2) sequences.

We created two synthetic read data sets from the two sets of single-cell genomes, setting the mean coverage to 1 for each genome, which yielded 392 790 reads for set 1 and 4 994 546 for set 2. We used `randomreads.sh` from the BBmap software suite with options `paired snprate=0.005 adderrors coverage=1 len=150 mininsert=150 maxinsert=350 gaussian=true` to simulate 2×150 bp paired-end overlapping reads with sequencing errors.

We then assembled the synthetic paired-end read data set with Megahit, metaSPAdes, Plass, SFA-SPA and Velvet, using the default parameters of each tool. We also tested Plass with with a stricter minimum sequeunce identity for merging sequences (option `--min-seq-id 97`, "Plass-97" in **Fig. 2**. For the nucleotide assemblers, we called proteins from the assembled contigs using Prodigal in metagenomics mode. We ignored all proteins shorter than 100 residues.

We calculated the *precision* by searching with the assembled proteins through the redundant reference set, using MMseqs2 with options `-a -s 5 --max-seqs 5000 --min-seq-id 0.89`. We filtered the aligned set by minimum sequence identity thresholds between 90% and 99%. For each search result, we only considered the longest alignment that fulfills the minimum sequence identity criterion. We computed the precision for each sequence identity threshold as the ratio of the total count of aligned residues divided by the total length of the assembled proteins. 100% precision is reached when all assembled protein residues can be aligned to a reference protein sequence.

We calculated the *sensitivity* by searching with the non-redundant reference set through the assembled proteins, using MMseqs2 with options `-a -s 5 --max-seqs 500000 --min-seq-id 0.89`. We filtered the aligned set by minimum sequence identity thresholds between 90% and 99%. For each search result, we only considered the longest alignment that fulfills the minimum sequence identity criterion. We computed the sensitivity for each sequence identity threshold as the ratio of the total count of aligned residues divided by the total length of the proteins in the non-redundant set. 100% sensitivity is reached when all reference protein residues can be aligned to an assembled protein sequence.

Accuracy of functional annotation of assembled proteins. To test the impact of chimeric assemblies and assembly quality on functional annotation quality, we measured the accuracy of functional annotations on the proteins assembled from the reads simulated from the single-cell marine genomes (**Fig. 2**). We functionally annotated each protein sequence assembled from the simulated reads and each reference protein from the single-cell genomes with an Orthologous Group using the eggNOG mapper[35] and the eggNOG database (version 4.5.1)[36] (options `-d bact -m diamond -override -cpu 16`). We compared the eggNOG annotation of each assembled protein sequence with the annotation of the best-matching sequence found in the reference protein set using MMseqs2. If the Orthologous Groups differ the annotation is false positive (FP), otherwise true positive (TP).

Protein sequence recovery on metagenomic datasets. For the benchmark test on real metagenomic data (**Fig. a-d**) we used the following datasets: (b) a single human gut sample from SRA (`SRR5024285`) [37], (c) 775 samples from Tara eukaryotic metatranscriptomes downloaded from the ENA (`PRJEB6609`) [38], (d) a soil sample from the IMG project 1003784 (sample: 6398.7.44014), (e) 12 samples from the same project (samples: 6679.7.51457 6478.6.45123, 6679.6.51456, 6398.7.44014, 6478.7.45124, 6674.6.51288, 6679.5.51455, 6674.4.51285, 6478.5.45122, 6478.4.45121, 6674.3.51284, 6674.5.51286). The soil data is also available at the NCBI Project PRJNA330082. All samples used in **Fig. 2b,d,e** consist of paired-end reads of 2 × 150 bp length, while **c** consists of reads with 2 × 102 bp length.

We assembled paired-end reads in datasets **b,d,e** using Megahit and Plass with default parameters. The benchmarks for sets in **Fig. 2b-d** were carried out on a single . The co-assembly in **Fig. 2e** was performed on a server with two 14-core Intel Xeon E5-2680 v4 CPUs with 768 GB RAM. During the co-assembly, Megahit aborted with a segmentation fault on the 768 GB server. We therefore performed twelve separate assemblies and pooled the results.

We could compare Plass only to Megahit on datasets **b,d,e**, since Velvet terminated with segmentation faults, metaSPAdes terminated with messages specifying a required amount of RAM in excess of the available 128 GB, and SFA-SPA did not finish execution within three days.

For **Fig. 2c**, we assembled the 775 Tara metatranscriptomes using Plass and compared the results with the Marine Atlas of Tara Oceans Unigene (MATOU) catalog [38], assembled using Velvet. For that purpose, we called protein sequences using Prodigal in metagenomics mode on all MATOU contigs, since these often do not contain full-length protein sequences. Eukaryotic protein sequences contain repeats more frequently than viral or prokayotic ones. We therefore masked low complexity regions of the assemblies created by Plass using tantan [39] and removed all assembled proteins with more than 50% masked residues.

To analyze the diversity of the obtained sets at various redundancy levels, we clustered all assembled protein sequence sets with Linclust using the parameters `--kmer-per-seq 80 --cluster-mode 2 --cov-mode 1 -c 0.9` at sequence identity thresholds `--min-seq-id` from 50% to 90%.

Taxonomic classification and quantification. We investigated the influence of the assembly method on the taxonomic composition (**Supplemental Fig. 5**). Instead of matching nucleotide reads to reference genomes, we here perform the taxonomic matching on the protein level because, first, many species sampled with metagenomics do not contain a close homolog in the reference databases, and second, protein-level comparison afford a much higher sensitivity to match to more distantly related sequences.

Our strategy is to (1) map reads – via the translated ORFs they contain – to assembled protein sequences and to (2) map the assembled protein sequences to taxonomic nodes in the NCBI taxonomic tree. We thereby map transitively each read to one taxonomic node.

To map the assembled protein sequences to taxonomic nodes (step 2 above), we implemented the 2bLCA protocol [40] as new MMseqs2 module `mmseqs taxonomy` (**Supplementary Fig. 4**) and assigned the assembled protein sequences to the 90% redundancy-filtered Uniprot database (Uniclust90 2017_07) [28], which contains taxonomic assignments to the NCBI tree for each sequence.

Using the two-step transitive mapping, we computed read counts for all taxonomic nodes. We then pooled the counts for each phylum in the tree and in addition recorded counts of reads assigned by 2bLCA to taxa above the phylum level. Only the 8 most abundant taxa were then kept, and counts of all others were pooled into a category "Others".

In **Supplementary Fig. 5a** we show the results for the soil sample assemblies from **c** (blue: Megahit, red: Plass) and the assemblies of the 12 soil samples from **d** (light blue: Megahit, light red: Plass), together with the ratios on top. The inset gives the fraction of reads in the single and the 12 soil samples that could be mapped to an assembled protein sequence with a minimum sequence identity of 90% (step 1 above).

In **Supplementary Fig. 5b** we show the count of assembled amino acids within various coverage ranges. Coverage of an assembled protein sequence is the sum of the number of residues aligned to that sequence during mapping divided by the length of the assembled protein sequence.

Around 5 to 10 times more reads can be mapped to the set of protein sequences assembled by Plass (red) than to the set predicted by Prodigal on the Megahit assembly. The gains are particularly high for high coverages.

Soil Reference Catalog assembly and analysis. For the Soil Reference Catalog (SRC), we downloaded from the sequence read archive (SRA) at the NCBI/NIH all 640 metagenomic datasets that (1) had the "soil metagenome" taxon identifier, (2) had dates between 01/2014 and 02/2018, (3) were sequenced on Illumina HiSeq or NovaSeq machines, and (4) had paired-end reads of at least 2 × 150 bp length. Sample identifiers are contained in a file `SRC_sample_ids.txt` at https://github.com/martin-steinegger/plass-analysis

Plass assembled the 18 Tbp of raw reads on a small cluster of servers with 2 × 8-core Intel Xeon E5-2640v3 CPUs and 128 GB RAM. We removed protein sequences shorter than 100 residues and redundancy-filtered the protein sequences from each sample using Linclust with options `--min-seq-id 0.95 --alignment-mode 3 -c 0.99 --cov-mode 1 --cluster-mode 2`). We pooled these 12 billion protein sequences and further reduced their redundancy by clustering with Linclust (`--cov-mode 1 -c 0.9 --min-seq-id 0.9`). The clustering was done hierarchically, since Linclust can only process $2^{32}-1$ sequences at once. The final set contains 2 022 891 389 sequences.

We chose two metagenomic soil sets (`SRR5919294` and `SRR6201924`) that were not part of the 640 datasets used for building the SRC. We merged overlapping read pairs using FLASH [23], sampled 100 000 merged reads per sample, predicted protein sequence fragments using Prodigal [34], and searched through the 90%-redundancy-filtered versions of SRC and the Uniprot database [28] using the `mmseqs map` workflow (see below). We computed the fraction of mapped reads out of the total read count while demanding a minimum sequence identity of 50% or 90% using the option `--min-seq-id`.

Read mapping. In this study, we use the novel `mmseqs map` workflow from the MMseqs2 package to find very similar protein sequence matches in a protein sequence database. It first calls the `mmseqs prefilter` module (with a low sensitivity setting of `-s 2`) to detect high scoring diagonals and then computes an ungapped alignment using the `mmseqs rescorediagonal` module. In contrast to the `mmseqs search` workflow, for maximum speed no gapped alignment is computed, query sequences are not masked for low complexity regions (`--mask-mode 0`), and no compositional bias correction is applied (`--comp-bias-corr 0`). By default, the mapping workflow requires that 90% of query sequence residues are aligned to a database sequence (`--cov-mode 2 -c 0.9`).

Software versions used. We used the following version of software in this article, Prodigal `V2.6.3`, FLASH `v1.2.11`, Velvet `1.2.10`, SFA-SPA `0.2.1`, metaSPAdes `v3.10.1`, Megahit `v1.1.1-2-g02102e1`, eggnog-mapper `1.0.3`.

Assembled protein sequence sets. The assembled protein sequence sets are available as FASTA formatted files at https://plass.mmseqs.org.

Code availability. Plass is GPLv3-licensed open source software. The source code and binaries for Plass can be downloaded at https://github.com/soedinglab/plass.

Data availability. All scripts and benchmark data including command-line parameters necessary to reproduce the benchmark and analysis results presented are available at https://github.com/martin-steinegger/plass-analysis.

[23] Magoc, T. & Salzberg, S. L. *Bioinformatics* **27**, 2957–2963 (2011).
[24] Ye, Y. & Tang, H. *J. Bioinform. Comput. Biol.* **7**, 455–471 (2009).
[25] Yang, Y. *et al.* *Bioinformatics* **31**, 1833–1835 (2015).
[26] Steinegger, M. & Söding, J. *Nat. Commun.* **9**, 2542 (2018).
[27] Sheetlin, S. *et al.* *Bioinformatics* **32**, 304–305 (2016).
[28] Mirdita, M. *et al.* *Nucleic Acids Res.* **45**, D170–D176 (2017).
[29] Kanehisa, M. *et al.* *Nucleic Acids Res.* **45**, D353–D361 (2016).
[30] Steinegger, M. & Söding, J. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
[31] Sczyrba, A. *et al.* *Nat. Methods* **14**, 1063–1071 (2017).
[32] Kashtan, N. *et al.* *Science* **344**, 416–420 (2014).
[33] Berube, P. M. *et al.* *Scientific Data* **5**, 180154 (2018).
[34] Hyatt, D. *et al.* *BMC Bioinformatics* **11**, 119 (2010).
[35] Huerta-Cepas, J. *et al.* *Mol Biol Evol* **34**, 2115–2122 (2017).
[36] Huerta-Cepas, J. *et al.* *Nucleic Acids Res* **44**, D286–93 (2016).
[37] Lee, S. T. M. *et al.* *Microbiome* **5**, 50 (2017).
[38] Carradec, Q. *et al.* *Nat. Commun.* **9**, 373 (2018).
[39] Frith, M. C. *Nucleic Acids Res.* **39**, e23 (2011).
[40] Hingamp, P. *et al.* *ISME J.* **7**, 1678–1695 (2013).

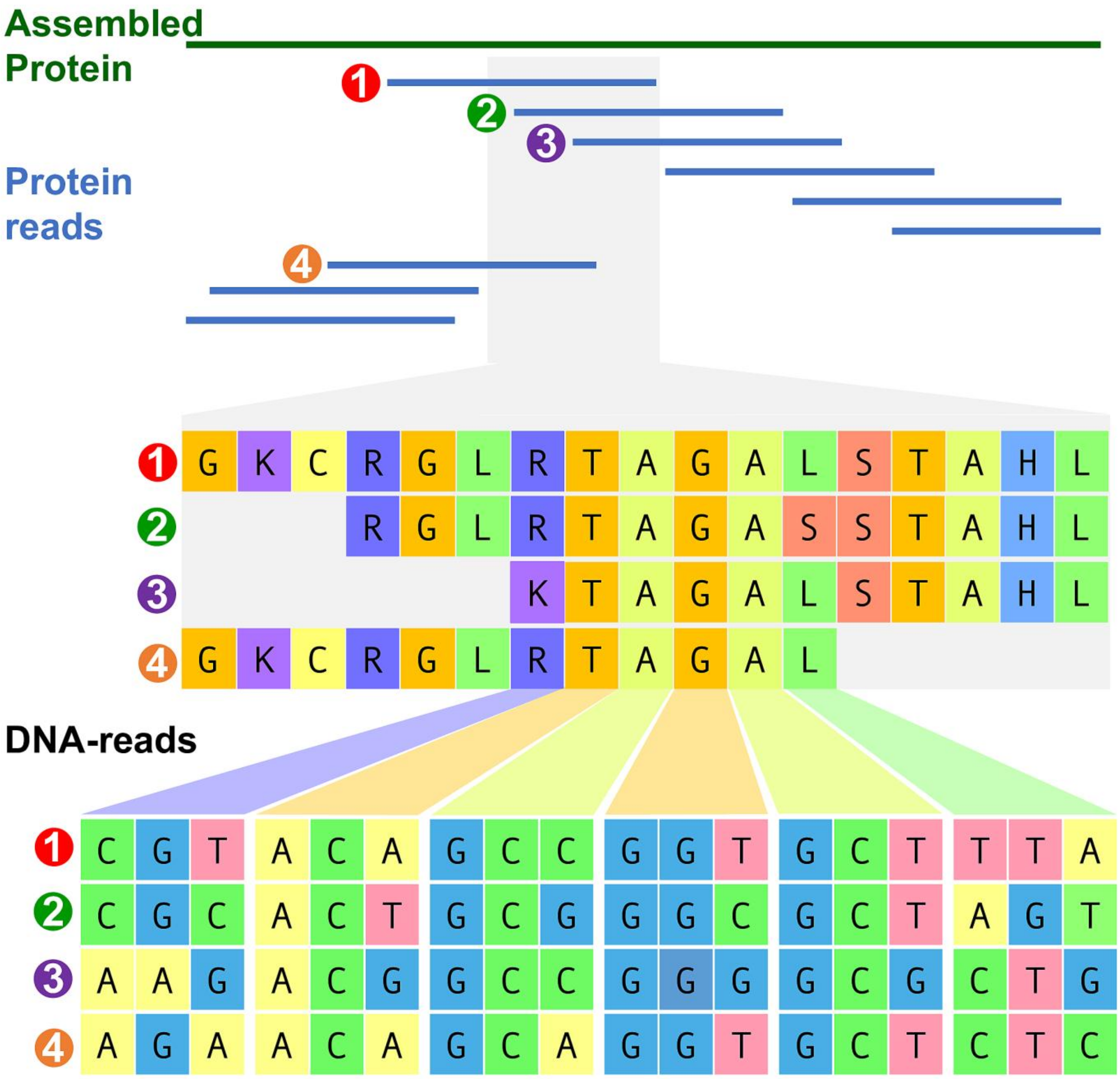
nature methods SUPPLEMENTARY INFORMATION
https://doi.org/10.1038/s41592-019-0437-4

In the format provided by the authors and unedited.

Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold

Martin Steinegger[1,2,3*], Milot Mirdita[1] and Johannes Söding[1*]

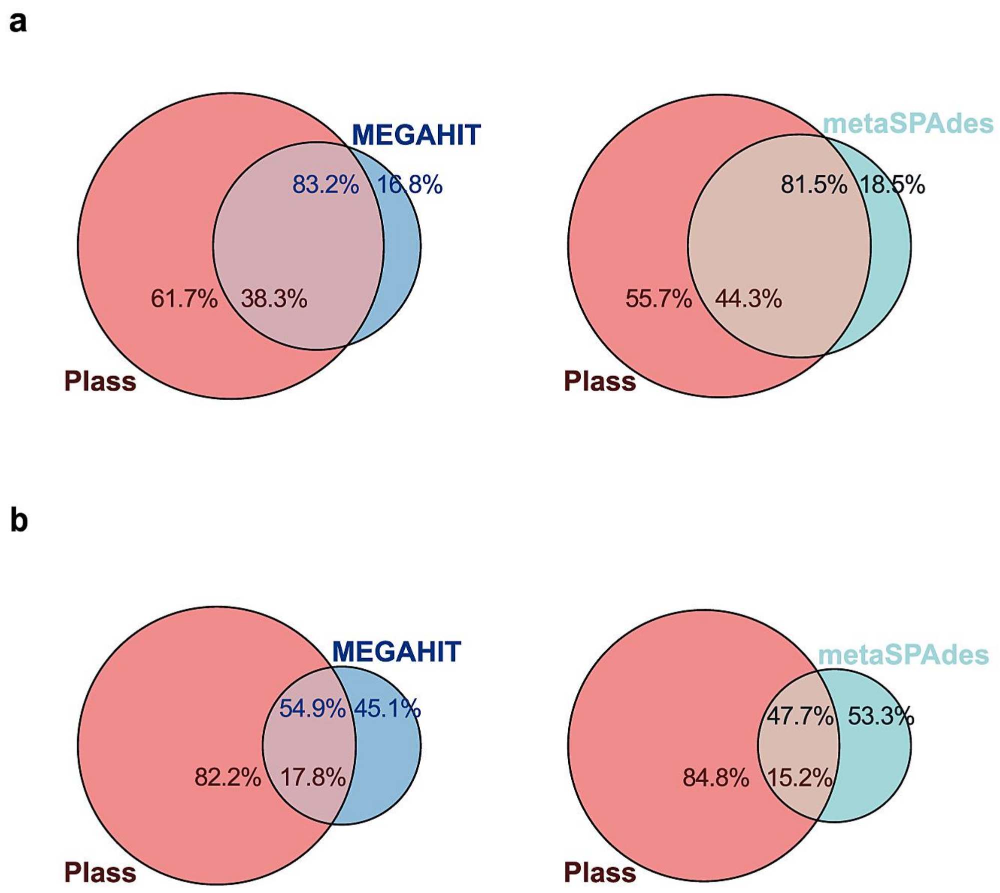
[1]Quantitative and Computational Biology Group, Max-Planck Institute for Biophysical Chemistry, Göttingen, Germany. [2]Department of Chemistry, Seoul National University, Seoul, Korea. [3]Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, MD, USA. *e-mail: martin.steinegger@mpibpc.mpg.de; soeding@mpibpc.mpg.de


Assembled Protein
Protein reads
1
2
3
4
1 G K C R G L R T A G A L S T A H L
2 R G L R T A G A S S T A H L
3 K T A G A L S T A H L
4 G K C R G L R T A G A L
DNA-reads
1 C G T A C A G C C G G T G C T T T A
2 C G C A C T G C G G G C G C T A G T
3 A A G A C G G C C G G G G C G C T G
4 A G A A C A G C A G G T G C T C T C

Supplementary Figure 1

Schematic comparison of a nucleotide- and a protein assembly.

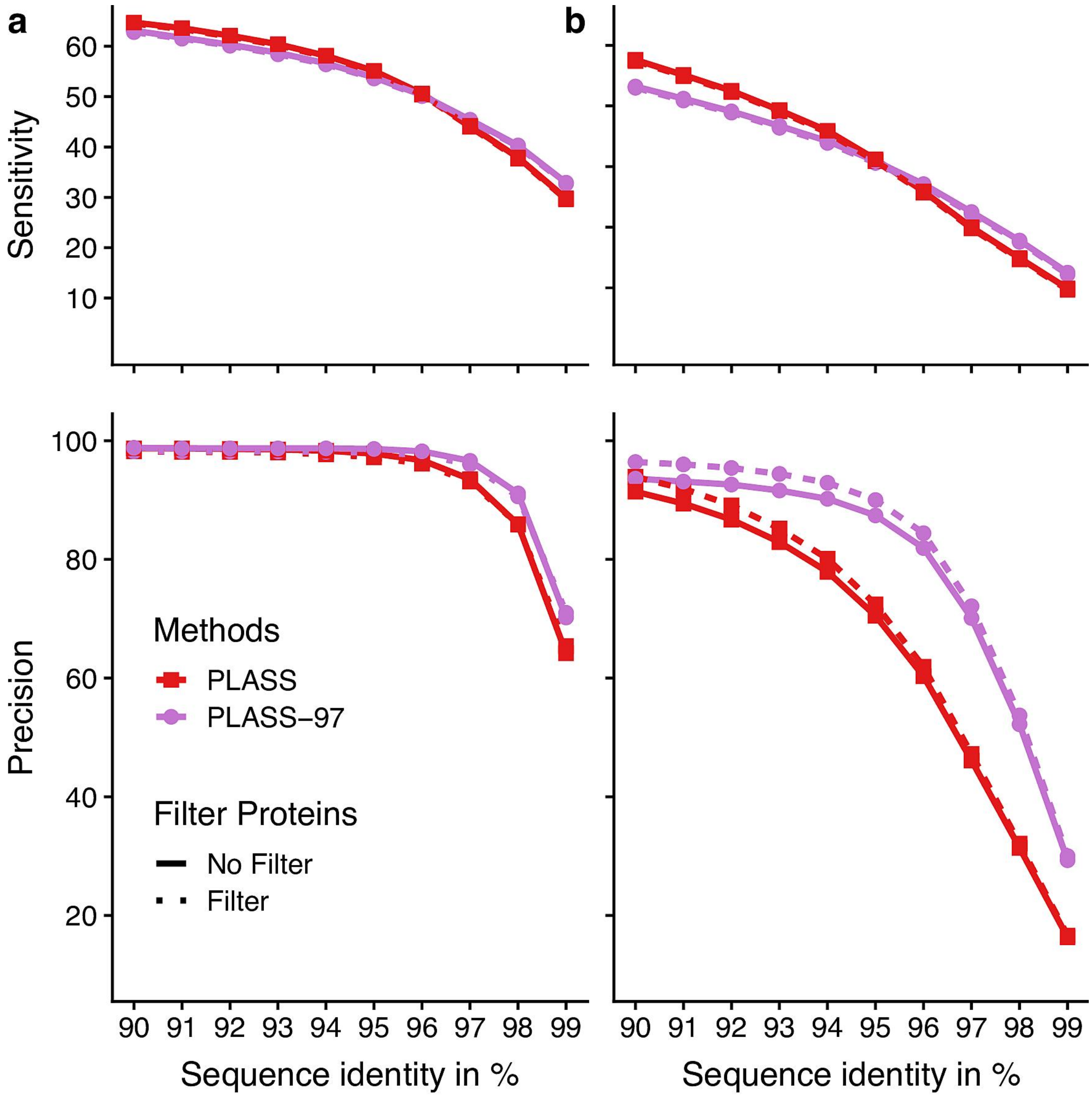
On top is the final protein assembly follow ed by the stacked overlapping protein reads. The small gray section highlights the multiple protein sequence alignment of the overlapping reads and below the respective nucleotide alignment. Less ambiguity is visible on the protein level due to conservative mutations (mutations w ith similar biochemical properties) compared to the nucleotide level, resulting in an assembly that is more robust to microdiversity in the population.



Supplementary Figure 2

Overlap of assemblies of Plass with Megahit and metaSPAdes

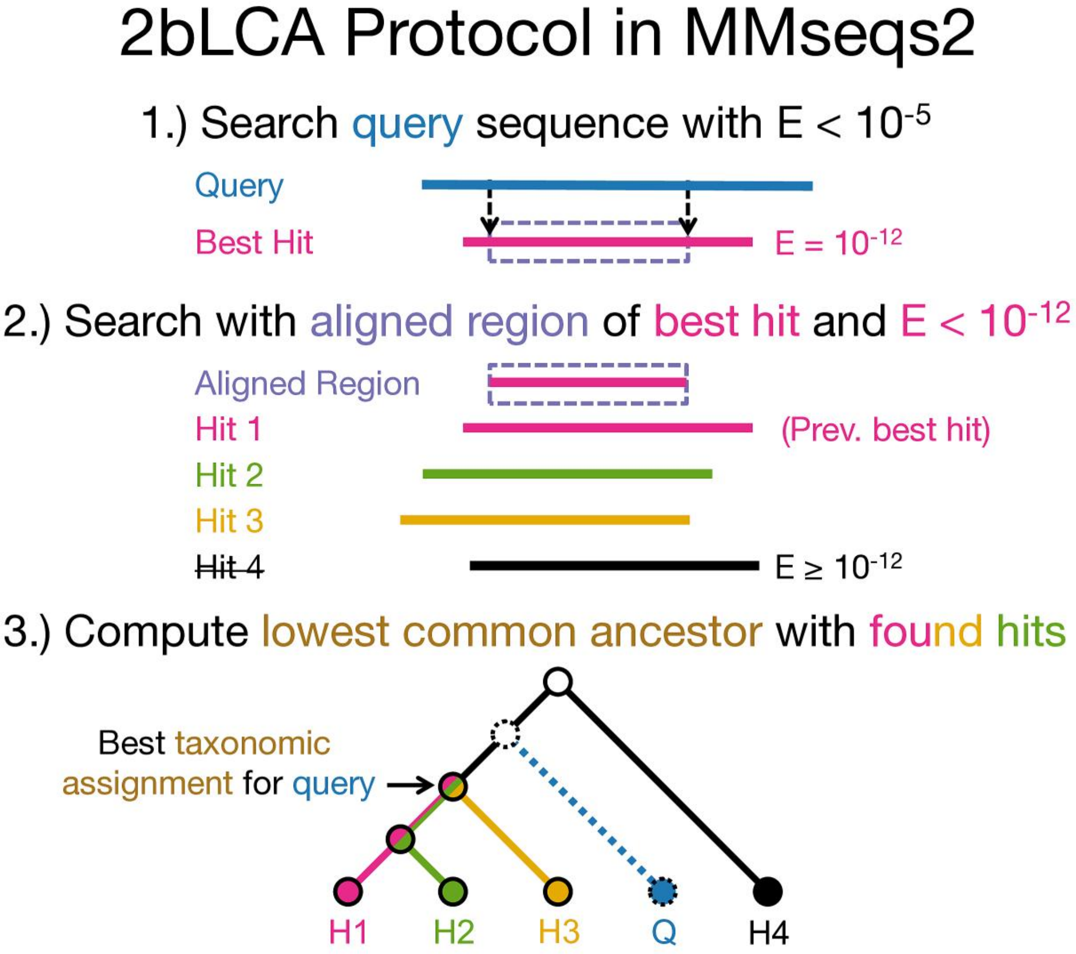
(a) left: A fraction 38.3% of amino acids in the Plass-assembled proteins of set 1 is covered by alignments to proteins in the Megahit assembly at a minimum sequence identity cut-off of 99%. Conversely, 83.2% of proteins in the Megahit-assembled set 1 is covered by alignments with Plass-assembled proteins. Right: Same as on the left but comparing the Plass assembly with the metaSPAdes assembly. (b) Same as (a) but for protein set 2



Supplementary Figure 3

Effect of neural network filter to remove wrong translation frames.

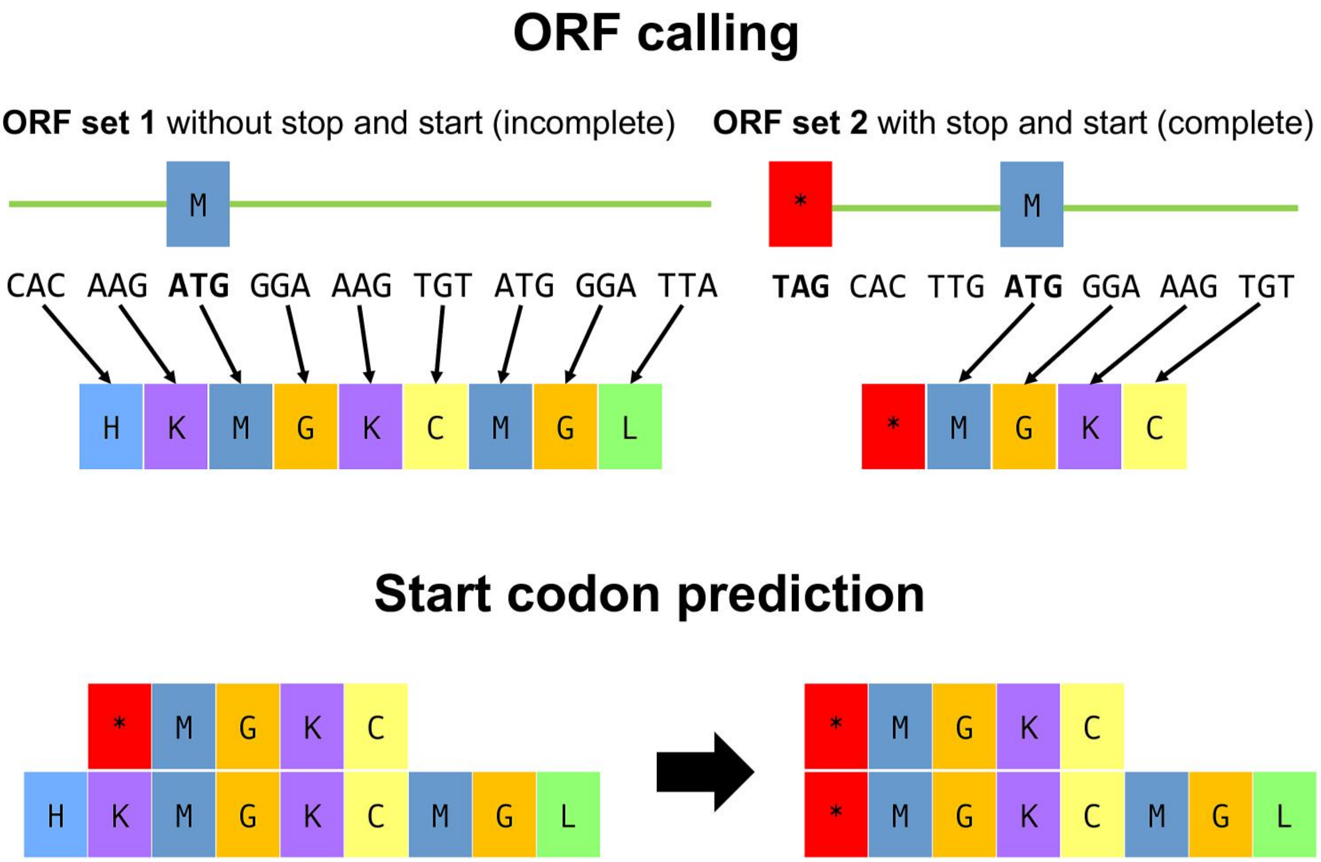
Sensitivity and precision in set 1 (a) and set 2 (b). Top: assembly sensitivity is the fraction of reference sequence amino acids that matches to an assembled protein sequence. Bottom: assembly precision is the fraction of assembled amino acids that matches to a reference protein at the minimum sequence identity on the x-axis. Plass uses a minimum sequence identity for merging fragments of 90%, Plass-97 uses a threshold of 97%.


2bLCA Protocol in MMseqs2
1.) Search query sequence with E < 10^{-5}
Query
Best Hit
E = 10^{-12}
2.) Search with aligned region of best hit and E < 10^{-12}
Aligned Region
Hit 1
(Prev. best hit)
Hit 2
Hit 3
~~Hit 4~~
E ≥ 10^{-12}
3.) Compute lowest common ancestor with found hits
Best taxonomic assignment for query
H1
H2
H3
Q
H4

Supplementary Figure 4

Comparison of Megahit assignment using the 2bLCA protocol

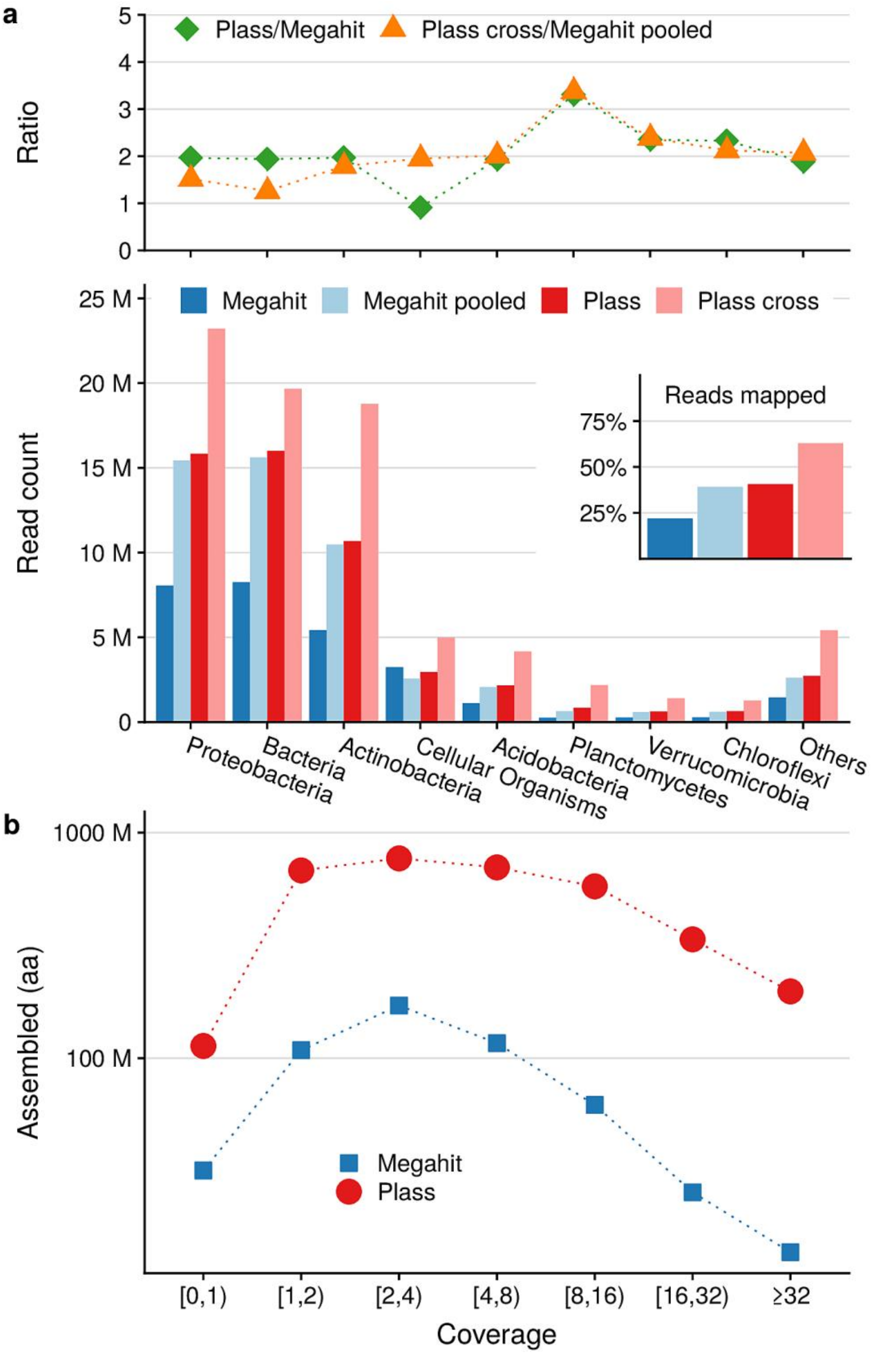
The MMseqs2 taxonomy assignment workflow uses three steps to assign a taxonomic label to a query sequence. (1) We search with the query sequence against a reference database and extract the aligned subsequence of the best hit. (2) This sequence is matched again against the reference database. Each hit with an E-value smaller than the best hit E-value from the previous search is accepted. (3) We compute the lowest common ancestor based on the taxonomic labels of all accepted hits.


ORF calling
ORF set 1 without stop and start (incomplete)
ORF set 2 with stop and start (complete)
M
*
M
CAC AAG ATG GGA AAG TGT ATG GGA TTA
TAG CAC TTG ATG GGA AAG TGT
H K M G K C M G L
* M G K C
Start codon prediction
* M G K C
H K M G K C M G L
* M G K C
* M G K C M G L

Supplementary Figure 5

Plass ORF extraction and start codon prediction (ORF calling)

Plass extracts two sets of ORFs. ORF set 1 contains all translated ORFs with at least 45 codons. ORF set 2 contains all translated ORFs with at least 20 codons starting with a putative `ATG` start codon that is the first `ATG` codon after a stop codon in the same frame. (Start codon prediction) Plass predicts start codons with a consensus method using a multiple sequence alignment of ORF set 1 and 2. Wherever at least 20% of all methionines in one column are marked by a prepended asterisk, it removes the preceding residues from all other sequences and prepends an asterisk to all sequences to mark the start



Supplementary Figure 6

Taxonomy evaluation of the soil metagenome assembly.
(a) We investigate the taxonomic composition of the 8 most abundant taxa (all other taxa are pooled in “Others”) in the soil assemblies from Fig. 2d (blue: Megahit, red: Plass) and the assemblies of the 12 soil samples from Fig. 2e (light blue: Megahit, light red: Plass). On top w e show the read count ratios betw een Plass and Megahit, for both the single and 12 soil assemblies. The inset gives the fraction of reads in the single and the 12 soil samples that could be mapped to an assembled protein sequence. (b) We show the count of assembled amino acids w ithin various coverage ranges for Megahit (blue) and Plass (red) in the single soil sample.

5.2 HH-suite3 for fast remote homology detection and deep protein annotation

Publication:

HH-suite3 for fast remote homology detection and deep protein annotation

M. Steinegger, M. Meier, **M. Mirdita**, H. Vöhringer, S. J. Haunsberger, J. Söding[†]

(†) corresponding author

BMC Bioinformatics (2019), 20, 473.
Cited 252 times since 09/2019.

Code and software availability

HH-suite3 is available as free open source software (GPLv3) at `github.com/soedinglab/hh-suite`.

Author contributions

M.S. & J.S. designed research, M.S. developed vectorized code and performed analyses, M.Meier refactored code, added features, fixed bugs and performed benchmarks, **M.Mirdita** added features, fixed bugs and maintains databases, H.V. implemented mmCIF support, S.H. optimized the MAC algorithm memory usage, M.S. and J.S. wrote the manuscript.

Steinegger *et al. BMC Bioinformatics* (2019) 20:473
https://doi.org/10.1186/s12859-019-3019-7

BMC Bioinformatics

SOFTWARE Open Access

HH-suite3 for fast remote homology detection and deep protein annotation

Martin Steinegger[1,2], Markus Meier[1], Milot Mirdita[1], Harald Vöhringer[1,3], Stephan J. Haunsberger[4] and Johannes Söding[1*]

Abstract

Background: HH-suite is a widely used open source software suite for sensitive sequence similarity searches and protein fold recognition. It is based on pairwise alignment of profile Hidden Markov models (HMMs), which represent multiple sequence alignments of homologous proteins.

Results: We developed a single-instruction multiple-data (SIMD) vectorized implementation of the Viterbi algorithm for profile HMM alignment and introduced various other speed-ups. These accelerated the search methods HHsearch by a factor 4 and HHblits by a factor 2 over the previous version 2.0.16. HHblits3 is ∼10× faster than PSI-BLAST and ∼20× faster than HMMER3. Jobs to perform HHsearch and HHblits searches with many query profile HMMs can be parallelized over cores and over cluster servers using OpenMP and message passing interface (MPI). The free, open-source, GPLv3-licensed software is available at https://github.com/soedinglab/hh-suite.

Conclusion: The added functionalities and increased speed of HHsearch and HHblits should facilitate their use in large-scale protein structure and function prediction, e.g. in metagenomics and genomics projects.

Keywords: Homology detection, Sequence search, Protein alignment, Algorithm, Profile HMM, SIMD, Functional annotation

Introduction

A sizeable fraction of proteins in genomics and metagenomics projects remain without annotation due to the lack of an identifiable, annotated homologous protein [1]. A high sensitivity in sequence similarity searches increases the chance of finding a homologous protein with an annotated function or a known structure from which the function or structure of the query protein can be inferred [2]. Therefore, to find template proteins for comparative protein structure modeling and for deep functional annotation, the most sensitive search tools such as HMMER [3, 4] and HHblits [5] are often used [6–9]. These tools can improve homology detection by aligning not only single sequences against other sequences, but using more information in form of multiple sequence alignments (MSAs) containing many homologous sequences. From the frequencies of amino acids in each column of the MSA, they calculate a 20 × length matrix of position-specific amino acid substitution scores, termed “sequence profile”.

A profile Hidden Markov Model (HMM) extends sequence profiles by augmenting the position-specific amino acid substitution scores with position-specific penalties for insertions and deletions. These can be estimated from the frequencies of insertions and deletions in the MSA. The added information improves the sensitivity of profile HMM-based methods like HHblits or HMMER3 over ones based on sequence profiles, such as PSI-BLAST [10].

Only few search tools represent both the query and the target proteins as sequence profiles built from MSAs of homologous proteins [11–14]. In contrast, HHblits / HHsearch represent both the query and the target proteins as profile HMMs. This makes them among the most sensitive tools for sequence similarity search and remote homology detection [5, 15].

In recent years, various sequence search tools have been developed that are up to four orders of magni-

*Correspondence: soeding@mpibpc.mpg.de
[1] Quantitative and Computational Biology Group, Max-Planck Institute for Biophysical Chemistry, Am Fassberg 11, 81379 Munich, Germany
Full list of author information is available at the end of the article

tude faster than BLAST [16–19]. This speed-up addresses the need to search massive amounts of environmental next-generation sequencing data against the ever-growing databases of annotated sequences. However, no homology can be found for many of these sequences even with sensitive methods, such as BLAST or MMseqs2 [19].

Genomics and metagenomics projects could annotate more sequence by adding HHblits searches through the PDB, Pfam and other profile databases to their pipelines [8]. Additional computation costs would be marginal, since the version of HHblits presented in this work runs 20 times faster than HMMER, the standard tool for Pfam [20] and InterPro [21] annotations.

In this work, our goal was to accelerate and parallelize various HH-suite algorithms with a focus on the most time-critical tools, HHblits and HHsearch. We applied data level parallelization using Advanced Vector Extension 2 (AVX2) or Streaming SIMD Extension 2 (SSE2) instructions, thread level parallelization using OpenMP, and parallelization across computers using MPI. Most important was the ample use of parallelization through SIMD arithmetic units present in all modern Intel, AMD and IBM CPUs, with which we achieved speed-ups per CPU core of a factor 2 to 4.

Methods

Overview of HH-suite

The software HH-suite contains the search tools HHsearch [15] and HHblits [5], and various utilities to build databases of MSAs or profile HMMs, to convert MSA formats, etc.

HHsearch aligns a profile HMM against a database of target profile HMMs. The search first aligns the query HMM with each of the target HMMs using the Viterbi dynamic programming algorithm, which finds the alignment with the maximum score. The E-value for the target HMM is calculated from the Viterbi score [5]. Target HMMs that reach sufficient significance to be reported are realigned using the Maximum Accuracy algorithm (MAC) [22]. This algorithm maximizes the expected number of correctly aligned pairs of residues minus a penalty between 0 and 1 (parameter `-mact`). Values near 0 produce greedy, long, nearly global alignments, values above 0.3 result in shorter, local alignments.

HHblits is an accelerated version of HHsearch that is fast enough to perform iterative searches through millions of profile HMMs, e.g. through the Uniclust profile HMM databases, generated by clustering the UniProt database into clusters of globally alignable sequences [23]. Analogously to PSI-BLAST and HMMER3, such iterative searches can be used to build MSAs by starting from a single query sequence. Sequences from matches to profile HMMs below some E-value threshold (e.g. 10^{-3}) are added to the query MSA for the next search iteration.

HHblits has a two-stage prefilter that reduces the number of database HMMs to be aligned with the slow Viterbi HMM-HMM alignment and MAC algorithms. For maximum speed, the target HMMs are represented in the prefilter as discretized sequences over a 219-letter alphabet in which each letter represents one of 219 archetypical profile columns. The two prefilter stages thus perform a profile-to-sequence alignment, first ungapped then gapped, using dynamic programming. Each stage filters away 95 to 99% of target HMMs.

Overview of changes from HH-suite version 2.0.16 to 3

Vectorized viterbi HMM-HMM alignment

Most of the speed-up was achieved by developing efficient SIMD code and removing branches in the pairwise Viterbi HMM alignment algorithm. The new implementation aligns 4 (using SSE2) or 8 (using AVX2) target HMMs in parallel to one query HMM.

Fast MAC HMM-HMM alignment

We accelerated the Forward-Backward algorithm that computes posterior probabilities for all residue pairs (i, j) to be aligned with each other. These probabilities are needed by the MAC alignment algorithm. We improved the speed of the Forward-Backward and MAC algorithms by removing branches at the innermost loops and optimizing the order of indices, which reduced the frequency of cache misses.

Memory reduction

We reduced the memory required during Viterbi HMM-HMM alignment by a factor of 1.5 for SSE2 and implemented AVX2 with only a 1.3 times increase, despite the need to keep scores for 4 (SSE2) or 8 (AVX2) target profile HMMs in memory instead of just one. This was done by keeping only the current row of the 5 scoring matrices in memory during the dynamic programming ("Memory reduction for backtracing and cell-off matrices" section), and by storing the 5 backtrace matrices, which previously required one byte per matrix cell, in a single backtrace matrix with one byte per cell ("From quadratic to linear memory for scoring matrices" section). We also reduced the memory consumption of the Forward-Backward and MAC alignment algorithms by a factor of two, by moving from storing posterior probabilities with type `double` to storing their logarithms using type `float`. In total, we reduced the required memory by roughly a factor 1.75 (when using SSE2) or 1.16 (when using AVX2).

Accelerating sequence filtering and profile computation

For maximum sensitivity, HHblits and HHsearch need to reduce the redundancy within the input MSA by removing sequences that have a sequence identity to another sequence in the MSA larger than a specified cutoff (90% by default) [15]. The redundancy filtering takes time $O(NL^2)$, where N is the number of MSA sequences and L the number of columns. It can be a runtime bottleneck for large MSAs, for example during iterative searches with HHblits. A more detailed explanation is given in "SIMD-based MSA redundancy filter" section.

Additionally, the calculation of the amino acid probabilities in the profile HMM columns from an MSA can become time-limiting. Its run time scales as $O(NL^2)$ because for each column it takes a time $\sim O(NL)$ to compute column-specific sequence weights based on the subalignment containing only the sequences that have no gap in that column.

We redesigned these two algorithms to use SIMD instructions and optimized memory access through reordering of nested loops and array indices.

Secondary structure scoring

Search sensitivity could be slightly improved for remote homologs by modifying the weighting of the secondary structure alignment score with respect to profile column similarity score. In HH-suite3, the secondary structure score can contribute more than 20% of the total score. This increased the sensitivity to detect remote homologs slightly without negative impact on the high-precision.

New features, code refactoring, and bug fixes

HH-suite3 allows users to search a large number of query sequences by parallelizing HHblits/HHsearch searches over queries using OpenMP and MPI (`hhblits_omp`, `hhblits_mpi`, `hhsearch_omp`, `hhsearch_mpi`). We removed the limit on the maximum number of sequences in the MSAs (parameter `-maxseqs < max>`). We ported scripts in HH-suite from Perl to Python and added support for the new PDB format mmCIF, which we use to provide precomputed profile HMM and MSA databases for the protein data bank (PDB) [24], Pfam [20], SCOP [25], and clustered UniProt databases (Uniclust) [23].

We adopted a new format for HHblits databases in which the column state sequences used for prefiltering (former `*.cs219` files) are stored in the FFindex format. The FFindex format was already used in version 2.0.16 for the `a3m` MSA files and the `hhm` profile HMM files. This resulted in a ~4 s saving for reading the prefilter database and improved scaling of HHblits with the number of cores. We also integrated our discriminative, sequence context-sensitive method to calculate pseudocounts for the profile HMMs, which slightly improves sensitivities for fold-level homologies [26].

To keep HH-suite sustainable and expandable in the longer term, we extensively refactored code by improving code reuse with the help of new classes with inheritance, replacing POSIX threads (`pthreads`) with OpenMP parallelization, removing global variables, moving from `make` to `cmake`, and moving the HH-suite project to GitHub (https://github.com/soedinglab/hh-suite). We fixed various bugs such as memory leaks and segmentation faults occurring with newer compilers.

Supported platforms and hardware

HHblits is developed under Linux, tested under Linux and macOS, and should run under any Unix-like operating systems. Intel and AMD CPUs that offer AVX2 or at least SSE2 instruction sets are supported (Intel CPUs: since 2006, AMD: since 2011). PowerPC CPUs with AltiVec vector extensions are also supported.

Because we were unable to obtain funding for continued support of HH-suite, user support is unfortunately limited to bug fixes for the time being.

Parallelization by vectorization using SIMD instructions

All modern CPUs possess SIMD units, usually one per core, for performing arithmetic, logical and other operations on several data elements in parallel. In SSE2, four floating point operations are processed in a single clock cycle in dedicated 128-bit wide registers. Since 2012, the AVX standard allows to process eight floating point operations per clock cycle in parallel, held in 256 bit AVX registers. With the AVX2 extension came support for byte-, word- and integer-level operations, e.g. 32 single-byte numbers can be added or multiplied in parallel (32×1 byte $= 256$ bits). Intel has supported AVX2 since 2013, AMD since 2015.

HHblits 2.0.16 already used SSE2 in its prefilter for gapless and gapped profile-to-sequence alignment processing 16 dynamic programming cells in parallel, but it did not support HMM-HMM alignment using vectorized code.

Abstraction layer for SIMD-based vector programming

Intrinsic functions allow to write SIMD parallelized algorithms without using assembly instructions. However, they are tied to one specific variant of SIMD instruction set (such as AVX2), which makes them neither downwards compatible nor future-proof. To be able to compile our algorithms with different SIMD instruction set variants, we implemented an abstraction layer, `simd.h`. In this layer, the intrinsic functions are wrapped by preprocessor macros. Porting our code to a new SIMD standard therefore merely requires us to extend the abstraction layer to that new standard, whereas the algorithm remains unchanged.

The `simd.h` header supports SSE2, AVX2 and AVX-512 instruction sets. David Miller has graciously extended the `simd.h` abstraction layer to support the AltiVec vector extension of PowerPC CPUs. Algorithm 1 shows a function that computes the scalar product of two vectors.

Algorithm 1 Example C code for SIMD abstraction layer

```
float scalarProdIntrinsics(float* m1, float* m2, int n) {
   float prod = 0.0;
   __m128 Z = _mm_setzero_ps();
   for(int i = 0; i   n; i += 4) {
      __m128 X = _mm_load_ps(&m1[i]);
      __m128 Y = _mm_load_ps(&m2[i]);
      X = _mm_mul_ps(X, Y);
      Z = _mm_add_ps(X, Z);
   }
   for(int i = 0; i   4; i++)
      prod += _mm_extract_ps(Z, i);
   return prod;
}

float scalarProdAbstracted(float* m1, float* m2, int n) {
   float prod = 0.0;
   simd_float Z = simdf32_setzero();
   for(int i = 0; i   n; i += VECSIZE_FLOAT) {
      simd_float X = simdf32_load(&m1[i]);
      simd_float Y = simdf32_load(&m2[i]);
      X = simdf32_mul(X, Y);
      Z = simdf32_add(X, Z);
   }
   for(int i = 0; i   VECSIZE_FLOAT; i++)
      prod += simdf32_extract(Z, i);
   return prod;
}
```

Vectorized viterbi HMM-HMM alignments

The viterbi algorithm for aligning profile hMMs

The Viterbi algorithm, when applied to profile HMMs, is formally equivalent to global sequence alignment with position-specific gap penalties [27]. We had previously introduced a modification of the Viterbi algorithm that is formally equivalent to Smith-Waterman local sequence alignment [15]. In HH-suite we use it to compute the best-scoring local alignment between two profile HMMs.

HH-suite models MSA columns with $<$ 50% gaps (default value) by match states and all other columns as insertion states. By traversing through the states of a profile HMM, the HMM can "emit" sequences. A match state (M) emits amino acids according to the 20 probabilities of amino acids estimated from their fraction in the MSA column, plus some pseudocounts. Insert states (I) emit amino acids according to a standard amino acid background distribution, while delete states (D) do not emit any amino acids.

The alignment score between two HMMs in HH-suite is the sum over all co-emitted sequences of the log odds scores for the probability for the two aligned HMMs to co-emit this sequence divided by the probability of the sequence under the background model. Since M and I states emit amino acids and D states do not, M and I in one HMM can only be aligned with M or I states in the other HMM. Conversely, a D state can only be aligned with a D state or with a Gap G (Fig. 1). The co-emission score

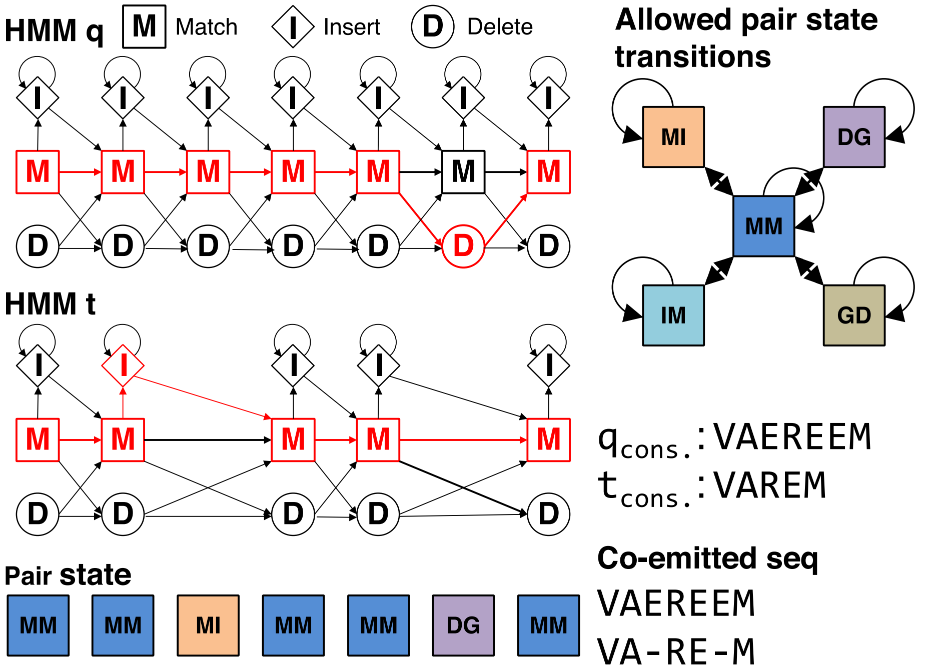


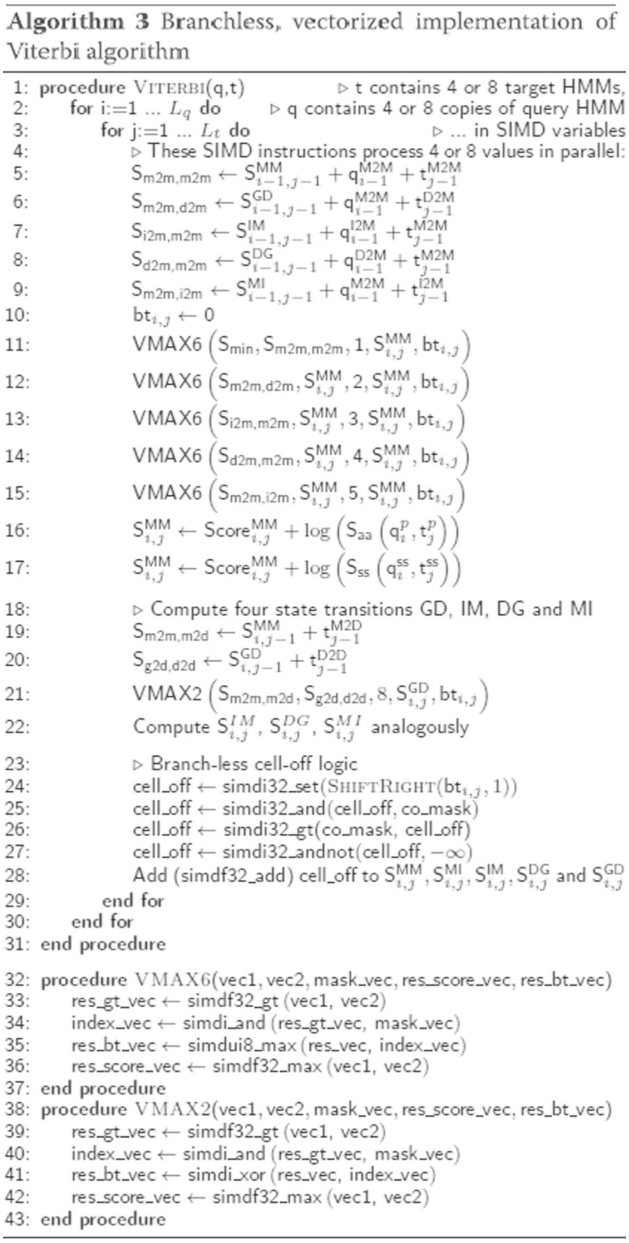
Fig. 1 HMM-HMM alignment of query and target. The alignment is represented as red path through both HMMs. The corresponding pair state sequence is MM, MM, MI, MM, MM, DG, MM

can be written as the sum of the similarity scores of the aligned profile columns, in other words the match-match (MM) pair states, minus the position-specific penalties for indels: delete-open, delete-extend, insert-open and insert-extend.

Algorithm 2 Viterbi algorithm for HMM-HMM alignment

```
procedure VITERBI(q,t)                                  ▷ query and target profile HMMs
    for i:=1 ... $L_q$ do                               ▷ $L_q$ = # match states of query HMM
        for j:=1 ... $L_t$ do                           ▷ $L_t$ = # match states of target HMM
            if cell_off$_{i,j}$ == true then                                ▷ cell forbidden?
                $S^{MM}_{i,j}, S^{MI}_{i,j}, S^{IM}_{i,j}, S^{DG}_{i,j}, S^{GD}_{i,j} \leftarrow -\infty$   ▷ Alignment cannot
                continue with next j                          ▷ pass through this cell
            end if

            MAX6( $S^{MM}_{i-1,j-1} + q^{M2M}_{i-1} + t^{M2M}_{j-1}$, ▷ $q^{M2M}_{i-1} = \log q_{i-1}(M,M)$
                  $S^{GD}_{i-1,j-1} + q^{M2M}_{i-1} + t^{D2M}_{j-1}$,
                  $S^{IM}_{i-1,j-1} + q^{I2M}_{i-1} + t^{M2M}_{j-1}$,
                  $S^{DG}_{i-1,j-1} + q^{D2M}_{i-1} + t^{M2M}_{j-1}$,
                  $S^{MI}_{i-1,j-1} + q^{M2M}_{i-1} + t^{I2M}_{j-1}$,
                  $S_{min}$,
                  $S^{MM}_{i,j}$, $bt^{MM}_{i,j}$)
            $S^{MM}_{i,j} \leftarrow S^{MM}_{i,j} + \log\left(S_{aa}\left(q^p_i, t^p_j\right)\right)$
            $S^{MM}_{i,j} \leftarrow S^{MM}_{i,j} + \log\left(S_{ss}\left(q^{ss}_i, t^{ss}_j\right)\right)$
            MAX2($S^{MM}_{i,j-1} + t^{M2D}_{j-1}, S^{GD}_{i,j-1} + t^{D2D}_{j-1}, S^{GD}_{i,j}, bt^{GD}_{i,j}$)
            ▷ Compute $S^{IM}_{i,j}$, $S^{DG}_{i,j}$, $S^{MI}_{i,j}$ analogously
            ...

        end for
    end for
end procedure

procedure MAX2(sMM, sXY, score, bt)
    if sMM > sXY then
        score ← sMM; bt ← MM
        ▷ The states STOP, MM, GD,... are 1-byte numbers
    else
        score ← sXY; bt ← SAME
    end if
end procedure

procedure MAX6(sSTOP, sMM, sGD, sIM, sDG, sMI, score, bt)
    if (sSTOP > sMM) then                      ▷ score will be max. score on return
        score ← sSTOP; bt ← STOP                          ▷ bt: for backtracing
    else
        score ← sMM; bt ← MM
    end if
    if (sGD > score) then
        score ← sGD; bt ← GD
    end if
    if (sIM > score) then
        score ← sIM; bt ← IM
    end if
    if (sDG > score) then
        score ← sDG; bt ← DG
    end if
    if (sMI > score) then
        score ← sMI; bt ← MI
    end if
end procedure
```

We denote the alignment pair states as MM, MI, IM, II, DD, DG, and GD. Figure 1 shows an example of two aligned profile HMMs. In the third column HMM *q* emits a residue from its M state and HMM *p* emits a residue from the I state. The pair state for this alignment column is MI. In column six of the alignment HMM *q* does not emit anything since it passes through the D state. HMM *p* does not emit anything either since it has a gap in the alignment. The corresponding pair state is DG. To speed up the alignment, we exclude pair states II and DD, and we only allow transitions between a pair state and itself and between pair state MM and pair states MI, IM, DG, or GD.



Algorithm 3 Branchless, vectorized implementation of Viterbi algorithm

```
procedure VITERBI(q,t)                          ▷ t contains 4 or 8 target HMMs,
    for i:=1 ... $L_q$ do              ▷ q contains 4 or 8 copies of query HMM
        for j:=1 ... $L_t$ do                               ▷ ... in SIMD variables
            ▷ These SIMD instructions process 4 or 8 values in parallel:
            $S_{m2m,m2m} \leftarrow S^{MM}_{i-1,j-1} + q^{M2M}_{i-1} + t^{M2M}_{j-1}$
            $S_{m2m,d2m} \leftarrow S^{GD}_{i-1,j-1} + q^{M2M}_{i-1} + t^{D2M}_{j-1}$
            $S_{i2m,m2m} \leftarrow S^{IM}_{i-1,j-1} + q^{I2M}_{i-1} + t^{M2M}_{j-1}$
            $S_{d2m,m2m} \leftarrow S^{DG}_{i-1,j-1} + q^{D2M}_{i-1} + t^{M2M}_{j-1}$
            $S_{m2m,i2m} \leftarrow S^{MI}_{i-1,j-1} + q^{M2M}_{i-1} + t^{I2M}_{j-1}$
            $bt_{i,j} \leftarrow 0$
            VMAX6 $\left(S_{min}, S_{m2m,m2m}, 1, S^{MM}_{i,j}, bt_{i,j}\right)$
            VMAX6 $\left(S_{m2m,d2m}, S^{MM}_{i,j}, 2, S^{MM}_{i,j}, bt_{i,j}\right)$
            VMAX6 $\left(S_{i2m,m2m}, S^{MM}_{i,j}, 3, S^{MM}_{i,j}, bt_{i,j}\right)$
            VMAX6 $\left(S_{d2m,m2m}, S^{MM}_{i,j}, 4, S^{MM}_{i,j}, bt_{i,j}\right)$
            VMAX6 $\left(S_{m2m,i2m}, S^{MM}_{i,j}, 5, S^{MM}_{i,j}, bt_{i,j}\right)$
            $S^{MM}_{i,j} \leftarrow Score^{MM}_{i,j} + \log\left(S_{aa}\left(q^p_i, t^p_j\right)\right)$
            $S^{MM}_{i,j} \leftarrow Score^{MM}_{i,j} + \log\left(S_{ss}\left(q^{ss}_i, t^{ss}_j\right)\right)$

            ▷ Compute four state transitions GD, IM, DG and MI
            $S_{m2m,m2d} \leftarrow S^{MM}_{i,j-1} + t^{M2D}_{j-1}$
            $S_{g2d,d2d} \leftarrow S^{GD}_{i,j-1} + t^{D2D}_{j-1}$
            VMAX2 $\left(S_{m2m,m2d}, S_{g2d,d2d}, 8, S^{GD}_{i,j}, bt_{i,j}\right)$
            Compute $S^{IM}_{i,j}$, $S^{DG}_{i,j}$, $S^{MI}_{i,j}$ analogously

            ▷ Branch-less cell-off logic
            cell_off ← simdi32_set(SHIFTRIGHT($bt_{i,j}$, 1))
            cell_off ← simdi32_and(cell_off, co_mask)
            cell_off ← simdi32_gt(co_mask, cell_off)
            cell_off ← simdi32_andnot(cell_off, $-\infty$)
            Add (simdf32_add) cell_off to $S^{MM}_{i,j}, S^{MI}_{i,j}, S^{IM}_{i,j}, S^{DG}_{i,j}$ and $S^{GD}_{i,j}$
        end for
    end for
end procedure

procedure VMAX6(vec1, vec2, mask_vec, res_score_vec, res_bt_vec)
    res_gt_vec ← simdf32_gt (vec1, vec2)
    index_vec ← simdi_and (res_gt_vec, mask_vec)
    res_bt_vec ← simdui8_max (res_vec, index_vec)
    res_score_vec ← simdf32_max (vec1, vec2)
end procedure
procedure VMAX2(vec1, vec2, mask_vec, res_score_vec, res_bt_vec)
    res_gt_vec ← simdf32_gt (vec1, vec2)
    index_vec ← simdi_and (res_gt_vec, mask_vec)
    res_bt_vec ← simdi_xor (res_vec, index_vec)
    res_score_vec ← simdf32_max (vec1, vec2)
end procedure
```

To calculate the local alignment score, we need five dynamic programming matrices S_{XY}, one for each pair state XY ∈ {MM, MI, IM, DG, GD}. They contain the score of the best partial alignment which ends in column i of q and column j of p in pair state XY. These five matrices are calculated recursively.

$$S_{\mathrm{MM}}\left(i,j\right) = S_{\mathrm{aa}}\left(q_i^p, t_j^p\right) + S_{\mathrm{ss}}\left(q_i^{ss}, t_j^{ss}\right) + \max\begin{cases} 0 \text{ (for } \textit{local} \text{ alignment)} \\ S_{\mathrm{MM}}(i-1,j-1) + \log\left(q_{i-1}(\mathrm{M,M})\ t_{j-1}(\mathrm{M,M})\right) \\ S_{\mathrm{MI}}(i-1,j-1) \ + \log\left(q_{i-1}(\mathrm{M,M})\ t_{j-1}(\mathrm{I,M})\right) \\ S_{\mathrm{II}}(i-1,j-1) \quad + \log\left(q_{i-1}(\mathrm{I,M})\ t_{j-1}(\mathrm{M,M})\right) \\ S_{\mathrm{DG}}(i-1,j-1) \ + \log\left(q_{i-1}(\mathrm{D,M})\ t_{j-1}(\mathrm{M,M})\right) \\ S_{\mathrm{GD}}(i-1,j-1) \ + \log\left(q_{i-1}\ (\mathrm{M,M})\ \ t_{j-1}(\mathrm{D,M})\right) \end{cases} \tag{1}$$

$$S_{\mathrm{MI}}\left(i,j\right) = \max\begin{cases} S_{\mathrm{MM}}(i-1,j) + \log\left(q_{i-1}(\mathrm{M,M})\ t_j(\mathrm{D,D})\right) \\ \ S_{\mathrm{MI}}(i-1,j) + \log\left(q_{i-1}(\mathrm{M,M})\ t_j(\mathrm{I,I})\right) \end{cases} \tag{2}$$

$$S_{\mathrm{DG}}\left(i,j\right) = \max\begin{cases} S_{\mathrm{MM}}(i-1,j) + \log\left(q_{i-1}(\mathrm{D,M})\right) \\ S_{\mathrm{DG}}(i-1,j) + \log\left(q_{i-1}(\mathrm{D,D})\right) \end{cases} \tag{3}$$

$$S_{aa}\left(q_i^p, t_j^p\right) = \log\sum_{a=1}^{20}\frac{q_i^p(a)\ t_j^p(a)}{f_a} \tag{4}$$

Vector q_i^p contains the 20 amino acid probabilities of q at position i, t_j^p are the amino acid probabilities t at j, and f_a denotes the background frequency of amino acid a. The score S_{aa} measures the similarity of amino acid distributions in the two columns i and j. S_{ss} can optionally be added to S_{aa}. It measures the similarity of the secondary structure states of query and target HMM at i and j [15].

Vectorizations of smith-Waterman sequence alignment

Much effort has gone into accelerating the dynamic programming based Smith-Waterman algorithm (at an unchanged time complexity of $O(L_q L_t)$). While substantial accelerations using general purpose graphics processing units (GPGPUs) and field programmable gated arrays (FPGAs) were demonstrated [28–31], the need for a powerful GPGPU and the lack of of a single standard (e.g. Nvidia's proprietary CUDA versus the OpenCL standard) have been impediments. SIMD implementations using the SSE2 and AVX2 standards with on-CPU SIMD vector units have demonstrated similar speed-ups as GPGPU implementations and have become widely used [3, 4, 32–35].

To speed up the dynamic programming (DP) using SIMD, multiple cells in the DP matrix are processed jointly. However the value in cell (i,j) depends on those in the preceding cells $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$. This data dependency makes acceleration of the algorithm challenging.

Four main approaches have been developed to address this challenge: (1) parallelizing over anti-diagonal stretches of cells in the DP matrices $((i,j), (i+1, j-1), \ldots (i+15, j-15)$, assuming 16 cells fit into one SIMD register) [32], (2) parallelizing over vertical or horizontal segments of the DP matrices (e.g. $(i,j), (i+1,j), \ldots (i+15,j))$ [33], (3) parallelizing over stripes of the DP matrices $((i,j), (i+1 \times D, j), \ldots (i+15 \times D, j)$ where D := `ceil`(query_length/16)) [34] and (4) where 16 cells (i,j) of 16 target sequences are processed in parallel [35].

The last option is the fastest method for sequence-sequence alignments, because it avoids data dependencies. Here we present an implementation of this option that can align one query profile HMM to 4 (SSE2) or 8 (AVX2) target profile HMMs in parallel.

Vectorized viterbi algorithm for aligning profile HMMs

Algorithm 2 shows the scalar version of the Viterbi algorithm for pairwise profile HMM alignment based on the iterative update Eqs. (1)–(3). Algorithm 3 presents our vectorized and branch-less version (Fig. 2). It aligns batches of 4 or 8 target HMMs together, depending on how many scores of type `float` fit into one SIMD register (4 for SSE2, 8 for AVX).

The vectorized algorithm needs to access the state transition and amino acid emission probabilities for these 4 or 8 targets at the same time. The memory is laid out (Fig. 3), such that the emission and transition probabilities of 4 or 8 targets are stored consecutively in memory. In this way, one set of 4 or 8 transition probabilities (for example MM) of the 4 or 8 target HMMs being aligned can be loaded jointly into one SIMD register.

The scalar versions of the functions `MAX6`, `MAX2` contain branches. Branched code can considerably slow down code execution due to the high cost of branch mispredictions, when the partially executed instruction pipeline has to be discarded to resume execution of the correct branch.

The functions `MAX6` and `MAX2` find the maximum score out of two or six input scores and also return the pair transition state that contributed the highest score. This state is stored in the backtrace matrix, which is needed to reconstruct the best-scoring alignment once all five DP matrices have been computed.

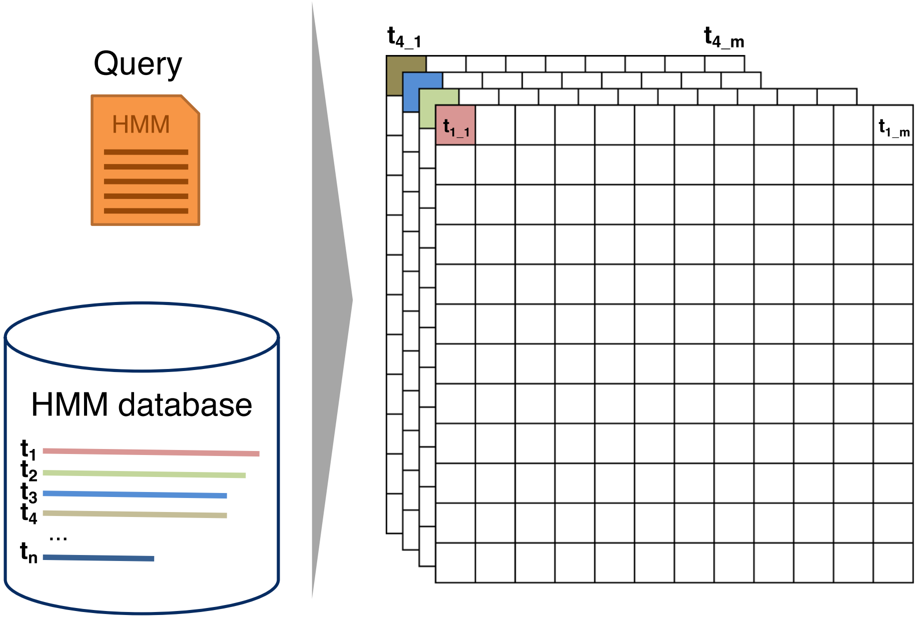


Fig. 2 SIMD parallelization over target profile HMMs. Batches of 4 or 8 database profile HMMs are aligned together by the vectorized Viterbi algorithm. Each cell (*i*,*j*) in the dynamic programming matrix is processed in parallel for 4 or 8 target HMMs

Algorithm 4 The similarity scores (Eq. (4)) for 4 or 8 target HMMs can be computed in parallel by 39 SIMD vector instructions in just 39 CPU clock cycles.

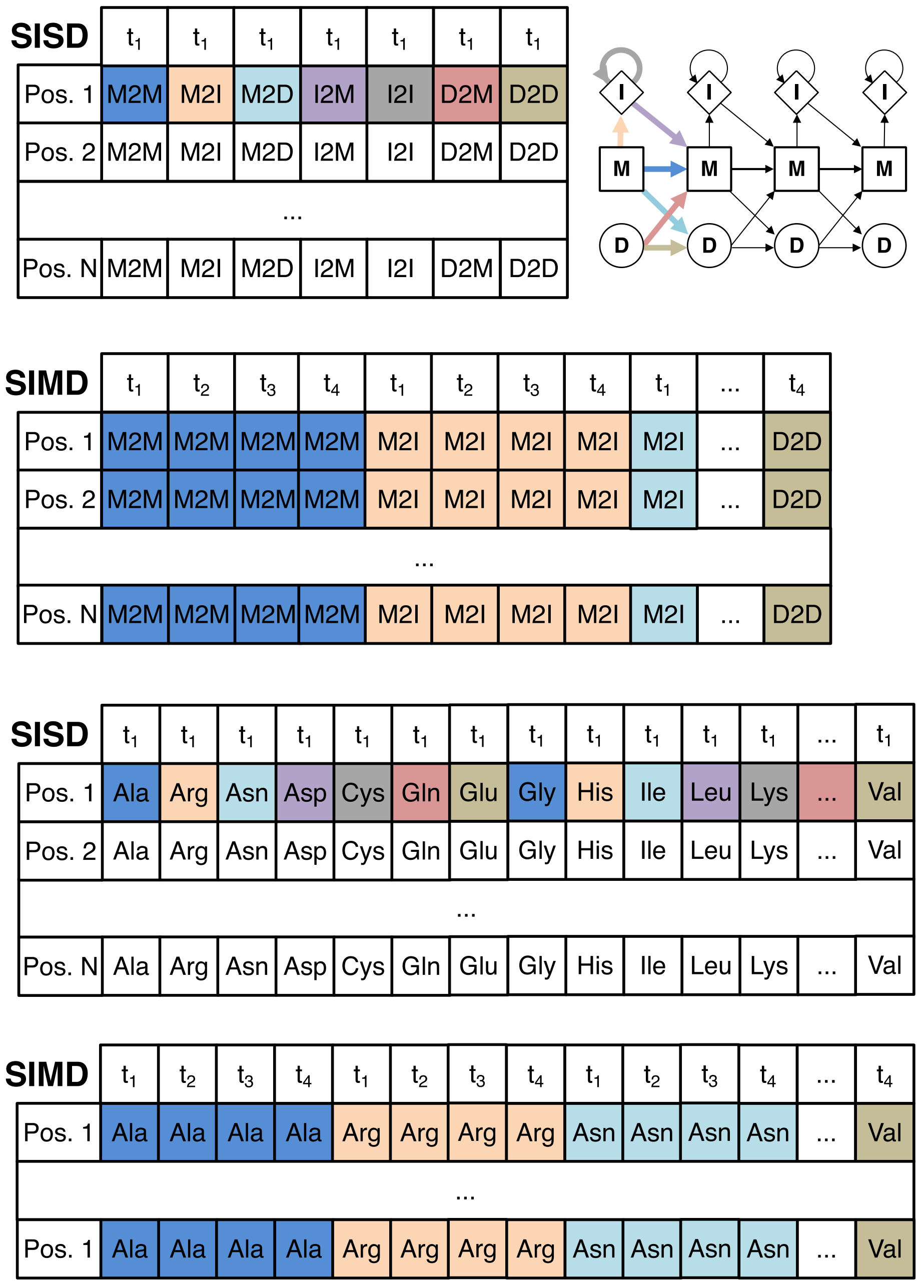
```
procedure $S_{aa}$SCALAR$(q, t)$                    ▷ Two profiles q and t
    res ← 0
    for i:=0 ... 19 do
        res ← res + (q$_i$ ∗ t$_i$)
    end for
    return res
end procedure

procedure $S_{aa}(q, t)$        ▷ Two SIMD-batches of profiles, q and t
    vector res0 ← t[0] ∗ q[0]
    vector res1 ← t[1] ∗ q[1]
    vector res2 ← t[2] ∗ q[2]
    vector res3 ← t[3] ∗ q[3]
    for  i:=4 ... 19 by 4 do
        res0 ← t[$i$] ∗ q[$i$]) + res0
        res1 ← t[$i+1$] ∗ q[$i+1$]) + res1
        res2 ← t[$i+2$] ∗ q[$i+2$]) + res2
        res3 ← t[$i+3$] ∗ q[$i+3$]) + res3
    end for
    res0 ← res0 + res1
    res2 ← res2 + res3
    return res0 + res2
end procedure
```

To remove the five if-statement branches in MAX6, we implemented a macro VMAX6 that implements one if-statement at a time. VMAX6 needs to be called 5 times, instead of just once as MAX6, and each call compares the current best score with the next of the 6 scores and updates the state of the best score so far by maximization. At each VMAX6 call, the current best state is overwritten by the new state if it has a better score.

We call the function VMAX2 four times to update the four states GD, IM, DG and MI. The first line in VMAX2 compares the 4 or 8 values in SIMD register sMM with the corresponding values in register sXY and sets all bits of the four values in SIMD register res_gt_vec to 1 if the value in sMM is greater than the one in sXY and to 0 otherwise. The second line computes a bit-wise AND between the four values in res_gt_vec (either 0x00000000 or 0xFFFFFFFF) and the value for state MM. For those of the 4 or 8 sMM values that were greater than the corresponding sXY value, we obtain state MM in index_vec, for the others we get zero, which represents staying in the same state. The backtrace vector can then be combined using an XOR instruction.

In order to calculate suboptimal, alternative alignments, we forbid the suboptimal alignment to pass through any cell (*i*,*j*) that is within 40 cells from any of the cells of the better-scoring alignments. These forbidden cells are stored in a matrix cell_off[i][j] in the scalar version of the Viterbi algorithm. The first if-statement in Algorithm 2 ensures that these cells obtain a score of $-\infty$.

To reduce memory requirements in the vectorized version, the cell-off flag is stored in the most significant bit of the backtracing matrix (Fig. 5) (see "Memory reduction for backtracing and cell-off matrices" section). In the SIMD Viterbi algorithm, we shift the backtracing matrix cell-off bit to the right by one and load four 32bit (SSE2) or eight 64bit (AVX2) values into a SIMD register (line 23). We extract only the cell-off bits (line 24)



SISD	t_1	t_1	t_1	t_1	t_1	t_1	t_1
Pos. 1	M2M	M2I	M2D	I2M	I2I	D2M	D2D
Pos. 2	M2M	M2I	M2D	I2M	I2I	D2M	D2D
...							
Pos. N	M2M	M2I	M2D	I2M	I2I	D2M	D2D

SIMD	t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4	t_1	...	t_4
Pos. 1	M2M	M2M	M2M	M2M	M2I	M2I	M2I	M2I	M2I	...	D2D
Pos. 2	M2M	M2M	M2M	M2M	M2I	M2I	M2I	M2I	M2I	...	D2D
...											
Pos. N	M2M	M2M	M2M	M2M	M2I	M2I	M2I	M2I	M2I	...	D2D

SISD	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	...	t_1
Pos. 1	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	...	Val
Pos. 2	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	...	Val
...														
Pos. N	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	...	Val

SIMD	t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4	...	t_4
Pos. 1	Ala	Ala	Ala	Ala	Arg	Arg	Arg	Arg	Asn	Asn	Asn	Asn	...	Val
...														
Pos. 1	Ala	Ala	Ala	Ala	Arg	Arg	Arg	Arg	Asn	Asn	Asn	Asn	...	Val

Fig. 3 The layout of the log transition probabilities (top) and emission probabilities (bottom) in memory for single-instruction single data (SISD) and SIMD algorithms. For the SIMD algorithm, 4 (using SSE2) or 8 (using AVX 2) target profile HMMs (t1 – t4) are stored together in interleaved fashion: the 4 or 8 transition or emission values at position *i* in these HMMs are stored consecutively (indicated by the same color). In this way, a single cache line read of 64 bytes can fill four SSE2 or two AVX2 SIMD registers with 4 or 8 values each

by computing an AND between the `co_mask` and the `cell_off` register. We set elements in the register with `cell_off` bit to 0 and without to `0xFFFFFFFF` by comparing if `cell_mask` is greater than `cell_off` (line 25). On line 26, we set the 4 or 8 values in the SIMD register `cell_off` to $-\infty$ if their cell-off bit was set and otherwise to 0. After this we add the generated vector to all five scores (MM, MI, IM, DG and GD).

A small improvement in runtime was achieved by compiling both versions of the Viterbi method, one with and one without cell-off logic. For the first, optimal alignment, we call the version compiled without the cell off logic and for the alternative alignments the version with cell-off logic enabled. In C/C++, this can be done with preprocessor macros.

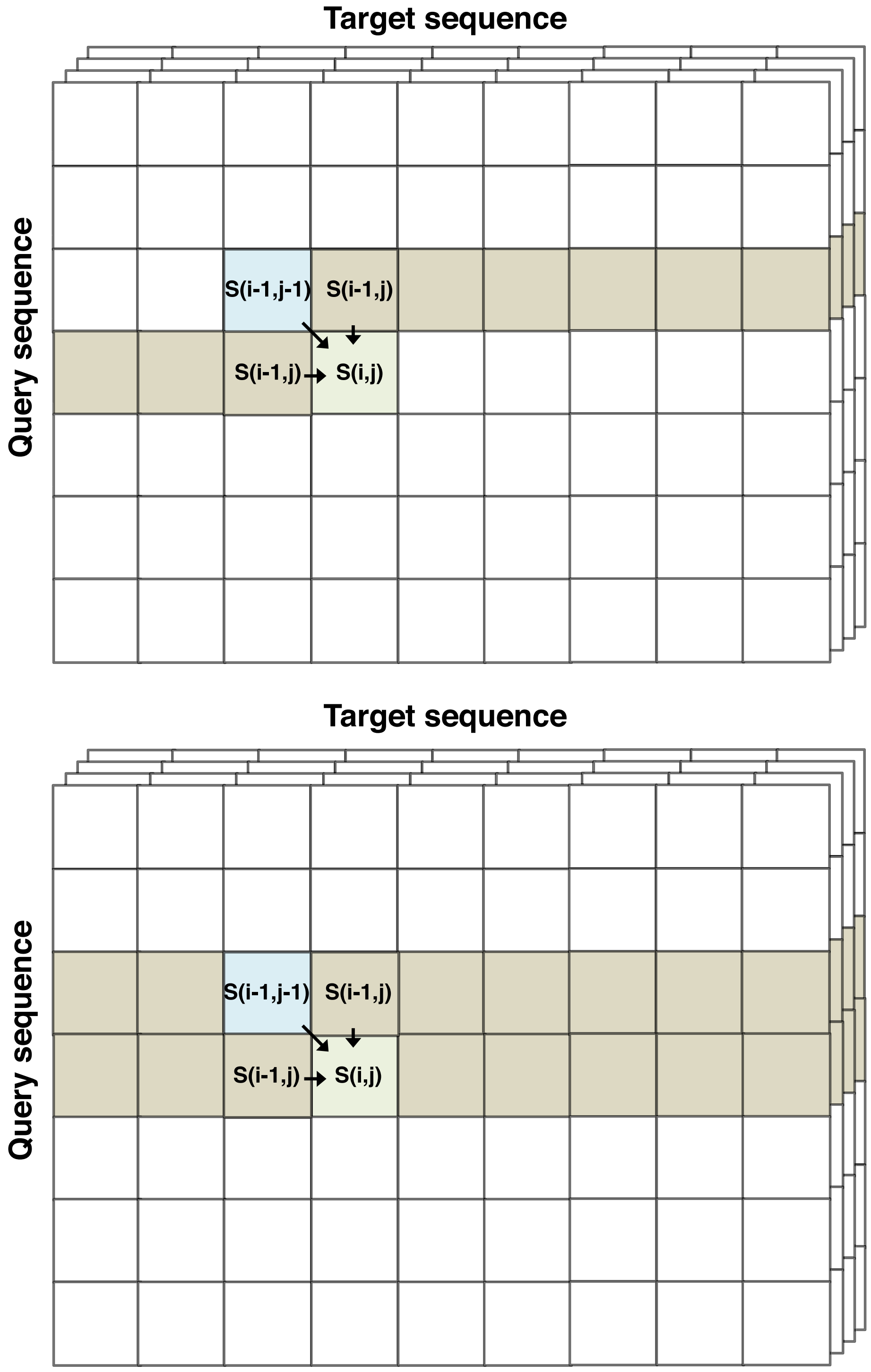


Fig. 4 Two approaches to reduce the memory requirement for the DP score matrices from $O(L_qL_t)$ to $O(L_t)$, where L_q and L_t are lengths of the query and target profile, respectively. (Top) One vector holds the scores of the previous row, $S_{XY}(i-1,\cdot)$, for pair state XY ∈{MM, MI, IM, GD and DG}, and the other holds the scores of the current row, $S_{XY}(i,\cdot)$ for pair state XY ∈{MM, MI, IM, GD and DG}. Vector pointers are swapped after each row has been processed. (Bottom) A single vector per pair state XY holds the scores of the current row up to $j-1$ and of the previous row for j to L_t. The second approach is somewhat faster and was chosen for HH-suite3

Shorter profile HMMs are padded with probabilities of zero up to the length of the longest profile HMM in the batch (Fig. 2). Therefore, the database needs to be sorted by decreasing profile HMM length. Sorting also improves IO performance due to linear access to the target HMMs for the Viterbi alignment, since the list of target HMMs that passed the prefilter is automatically sorted by length.

Vectorized column similarity score

The sum in the profile column similarity score S_{aa} in the first line in Algorithm 4 is is computed as the scalar product between the precomputed 20-dimensional vector $q_i^p(a)/f_a$ and $t_j^p(a)$. The SIMD code takes 39 instructions to compute the scores for 4 or 8 target columns, whereas the scalar version needed 39 instructions for a single target column.

From quadratic to linear memory for scoring matrices

Most of the memory in Algorithm 2 is needed for the five score matrices for pair states MM, MI, IM, GD and DG. For a protein of 15 000 residues, the five matrices need 15 000 × 15 000 × 4byte × 5 matrices = 4.5GB of memory per thread.

In a naive implementation, the vectorized algorithm would need a factor of 4 or 8 more memory than that, since it would need to store the scores of 4 or 8 target profile HMMs in the score matrices. This would require 36GB of memory per thread, or 576GB for commonly used 16 core servers.

However, we do not require the entire scoring matrices to reside in memory. We only need the backtracing matrices and the position $(i_{\text{best}}, j_{\text{best}})$ of the highest scoring cell to reconstruct the alignment.

We implemented two approaches. The first uses two vectors per pair state (Fig. 4 top). One holds the scores of the current row i, where (i, j) are the positions of the cell whose scores are to be computed, and the other vector holds the scores of the previous row $i - 1$. After all the scores of a row i have been calculated, the pointers to the vectors are swapped and the former row becomes the current one.

The second approach uses only a single vector (Fig. 4 bottom). Its elements from 1 to $j - 1$ hold the scores of the current row that have already been computed. Its elements from j to the last position L_t hold the scores from the previous row $i - 1$.

The second variant turned out to be faster, even though it executes more instructions in each iteration. However, profiling showed that this is more than compensated by fewer cache misses, probably owed to the factor two lower memory required.

We save a lot of memory by storing the currently needed scores of the target in a linear ring buffer of size $O(L_t)$. However, we still need to keep the backtracing matrix (see next subsection), of quadratic size $O(L_q L_t)$ in memory. Therefore the memory complexity remains unaffected.

Memory reduction for backtracing and cell-off matrices

To compute an alignment by backtracing from the cell $(i_{\text{best}}, j_{\text{best}})$ with maximum score, we need to store for each cell (i, j) and every pair state (MM, GD, MI, DG, IM) the previous cell and pair state the alignment would pass through, that is, which cell contributed the maximum score in (i, j). For that purpose it obviously suffices to only store the previous pair state.

HHblits 2.0.16 uses five different matrices of type `char`, one for each pair state, and one `char` matrix to hold the cell-off values (in total 6 bytes). The longest known protein Titin has about 33 000 amino acids. To keep a 33 000 × 33 000 × $6byte$ matrix in memory, we would need 6GB of memory. Since only a fraction of $\sim 10^{-5}$ sequences are sequences longer than 15 000 residues in the UniProt database, we restrict the default maximum sequence length to 15 000. This limit can be increased with the parameter `-maxres`.

But we would still need about 1.35GB to hold the backtrace and cell-off matrices. A naive SSE2 implementation would therefore need 5.4GB, and 10.8GB with AVX2. Because every thread needs its own backtracing and cell-off matrices, this can be a severe restriction.

Algorithm 5 Check if x,y have seq. identity > seqid_min

```
procedure FILTER(x, y, seqid_min)                      ▷ Two MSA sequences
    cov ← L; diff_min ← cov ∗ (1 − seqid_min)
    for i := 1 to L do
        if $x_i$ is gap OR $y_i$ is gap then
            cove ← cov − 1; diff_min ← cov ∗ (1 − seqid_min)
        else if $x_i$ not equal $y_i$ then
            diff ← diff + 1
            if diff >= diff_min then return 1   ▷ x, y dissimilar enough
            end if
        end if
    end for
    return 0                                            ▷ x,y too similar
end procedure
```

We reduce the memory requirements by storing all backtracing information and the cell-off flag in a single byte per cell (i, j). The preceding state for the IM, MI, GD, DG states can be held as single bit, with a 1 signifying that the preceding pair state was the same as the current one and 0 signifying it was MM. The preceding state for MM can be any of STOP, MM, IM, MI, GD, and DG. STOP represents the start of the alignment, which corresponds to the 0 in (eq. 1) contributing the largest of the 6 scores. We need three bits to store these six possible predecessor pair states. The backtracing information can, thus, be held in '4 + 3' bits, which leaves one bit for the cell-off flag (Fig. 5). Due to the reduction to one byte per cell we need only 0.9GB (with SSE2) or 1.8GB (with AVX2) per thread to hold the backtracing and cell-off information.

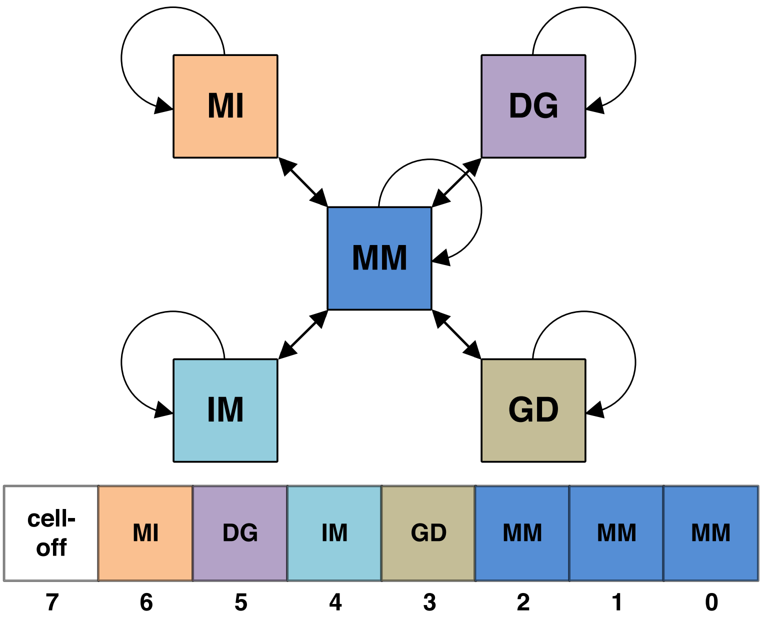


Fig. 5 Predecessor pair states for backtracing the Viterbi alignments are stored in a single byte of the backtrace matrix in HH-suite3 to reduce memory requirements. The bits 0 to 2 (blue) are used to store the predecessor state to the MM state, bits 3 to 6 store the predecessor of GD, IM, DG and MI pair states. The last bit denotes cells that are not allowed to be part of the suboptimal alignment because they are near to a cell that was part of a better-scoring alignment

Viterbi early termination criterion

For some query HMMs, a lot of non-homologous target HMMs pass the prefiltering stage, for example when they contain one of the very frequent coiled coil regions. To avoid having to align thousands of non-homologous target HMMs with the costly Viterbi algorithm, we introduced an early termination criterion in HHblits 2.0.16. We averaged 1/(1 + E-value) over the last 200 processed Viterbi alignments and skipped all further database HMMs when this average dropped below 0.01, indicating that the last 200 target HMMs produced very few Viterbi E-values below 1.

This criterion requires the targets to be processed by decreasing prefilter score, while our vectorized version of the Viterbi algorithm requires the database profile HMMs to be ordered by decreasing length. We solved this dilemma by sorting the list of target HMMs by decreasing prefilter score, splitting it into equal chunks (default size 2 000) with decreasing scores, and sorting target HMMs within each chunk by their lengths. After each chunk has been processed by the Viterbi algorithm, we compute the average of 1/(1 + E-value) for the chunk and terminate early when this number drops below 0.01.

SIMD-based MSA redundancy filter

To build a profile HMM from an MSA, HH-suite reduces the redundancy by filtering out sequences that have more than a fraction `seqid_max` of identical residues with another sequence in the MSA. The scalar version of the function (Algorithm 5) returns 1 if two sequences x and y have a sequence identity above `seqid_min` and 0 otherwise. The SIMD version (Algorithm 6) has no branches and processes the amino acids in chunks of 16 (SSE2) or 32 (AVX2). It is about ~11 times faster than the scalar version.

Algorithm 6 Vectorized version of Algorithm 5

```
procedure FILTERSIMD($x, y$, seqid_min)          ▷ Two MSA sequences
    cov ← L; diff_min ← cov ∗ (1 − seqid_min)
    $aa_{max}$ = _mm_set1_epi8(19)                  ▷ vector with 32 times 19
    for i := 1 to L & diff < diff_min do
        gaps_x ← _mm_cmpgt_epi8($x_i$ $aa_{max}$ ) ▷ pos's with gaps in x
        gaps_y ← _mm_cmpgt_epi8($y_i$, $aa_{max}$ ) ▷ pos's with gaps in y
        ▷ Compute mask (32 bit int) of positions with gap in x or y
        no_aa ← _mm_movemask_epi8(_mm_or_si128(gaps_x, gaps_y))
        ▷ Update number of aligned residues
        cov ← cov - CountBits(no_aa)
        diff_min ← cov ∗ (1 − seqid_min)
        ▷ Compute mask of positions with identical amino acids
        ident ← _mm_movemask_epi8 (_mm_cmpeq_epi8 ($x_i, y_i$))
        diff ← diff + 32 - CountBits(ident)
    end for
    return (diff >= diff_min)
end procedure
```

Results

Speed benchmarks

Speed of HHsearch 2.0.16 versus HHsearch 3

Typically more than 90% of the run time of HHsearch is spent in the Viterbi algorithm, while only a fraction of the time is spent in the maximum accuracy alignment. Only a small number of alignments reach an E-value low enough in the Viterbi algorithm to be processed further.

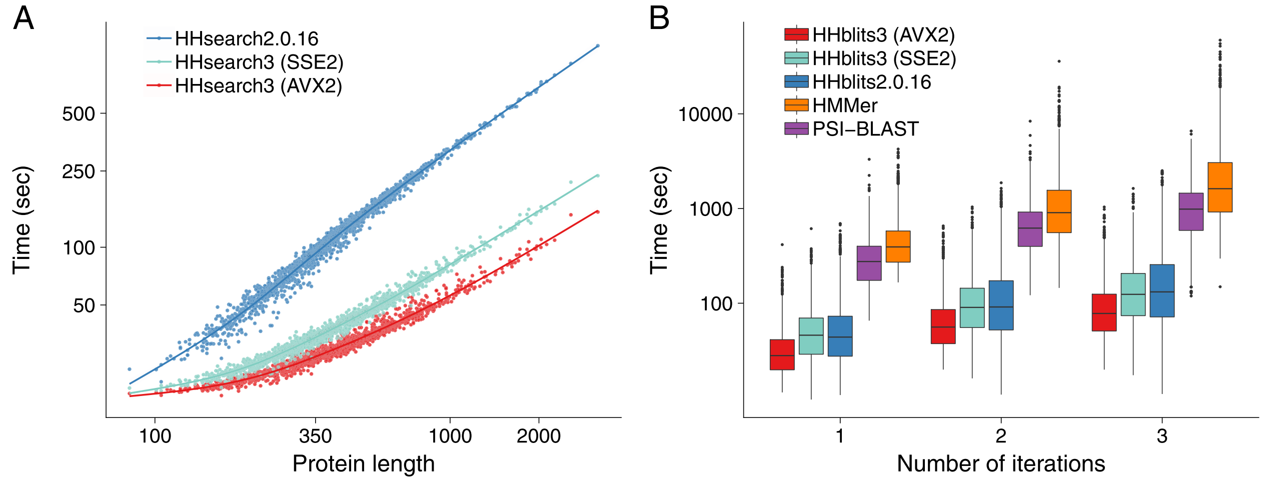


Fig. 6 Speed comparisons. **a** runtime versus query profile length for 1644 searches with profile HMMs randomly sampled from UniProt. These queries were searched against the PDB70 database containing 35 000 profile HMMs of average length 234. The average speedup over HHsearch 2.0.16 is 3.2-fold for SSE2- vectorized HHsearch and 4.2-fold for AVX2-vectorized HHsearch. **b** Box plot for the distribution of total runtimes (in logarithmic scale) for one, two, or three search iterations using the 1644 profile HMMs as queries. PSI-BLAST and HHMER3 searches were done against the UniProt database (version 2015_06) containing 49 293 307 sequences. HHblits searches against the uniprot20 database, a clustered version of UniProt containing profile HMMs for each of its 7 313 957 sequence clusters. Colored numbers: speed-up factors relative to HMMER3

HHsearch therefore profits considerably from the SIMD vectorization of the Viterbi algorithm.

To compare the speed of the HHsearch versions, we randomly selected 1 644 sequences from Uniprot (release 2015_06), built profile HMMs, and measured the total run time for searching with the 1644 query HMMs through the PDB70 database (version 05Sep15). The PDB70 contains profile HMMs for a representative set of sequences from the PDB [24], filtered with a maximum pairwise sequence identity of 70%. It contained 35 000 profile HMMs with an average length of 234 match states.

HHsearch with SSE2 is 3.2 times faster and HHsearch with AVX2 vectorization is 4.2 times faster than HHsearch 2.0.16, averaged over all 1644 searches (Fig. 6a). For proteins longer than 1000, the speed-up factors are 5.0 and 7.4, respectively. Due to a runtime overhead of ∼20 s that is independent of the query HMM length (e.g. for reading in the profile HMMs), the speed-up shrinks for shorter queries. Most of this speed-up is owed to the vectorization of the Viterbi algorithm: The SSE2-vectorized Viterbi code ran 4.2 times faster than the scalar version.

In HHblits, only part of the runtime is spent in the Viterbi algorithm, while the larger fraction is used by the prefilter, which was already SSE2-vectorized in HHblits 2.0.16. Hence we expected only a modest speed-up between HHblits 2.0.16 and SSE2-vectorized HHblits 3. Indeed, we observed an average speed-up of 1.2, 1.3, and 1.4 for 1, 2 and 3 search iterations, respectively (Fig. 6b), whereas AVX2-vectorized version is 1.9, 2.1, and 2.3 times faster than HHblits 2.0.16, respectively. AVX2-vectorized HHblits is 14, 20, and 29 times faster than HMMER3 [4] (version 3.1b2) and 9, 10, and 11 times faster than PSI-BLAST [10] (blastpgp 2.2.31) for 1, 2, and 3 search iterations.

All runtime measurements were performed using the Unix tool `time` on a single core of a computer with two Intel Xeon E5-2640v3 CPUs with 128GB RAM.

Sensitivity benchmark

To measure the sensitivity of search tools to detect remotely homologous protein sequences, we used a benchmarking procedure very similar to the one described in [5]. To annotate the uniprot20 (version 2015_06) with SCOP domains, we first generated a SCOP20 sequence set by redundancy-filtering the sequences in SCOP 1.75 [25] to 20% maximum pairwise sequence identity using `pdbfilter.pl` with minimum coverage of 90% from HH-suite, resulting in 6616 SCOP domain sequences. We annotated a subset of uniprot20 sequences by the presence of SCOP domains by searching with each sequence in the SCOP20 set with `blastpgp` through the consensus sequences of the uniprot20 database and annotated the best matching sequence that covered ≥ 90% of the SCOP sequence and that had a minimum sequence identity of at least 30%.

We searched with PSI-BLAST (2.2.31) and HMMER3 (v3.1b2) with three iterations, using the 6616 sequences in the SCOP20 set as queries, against a database made up of the UniProt plus the SCOP20 sequence set. We searched with HHblits versions 2.0.16 and 3 with three iterations

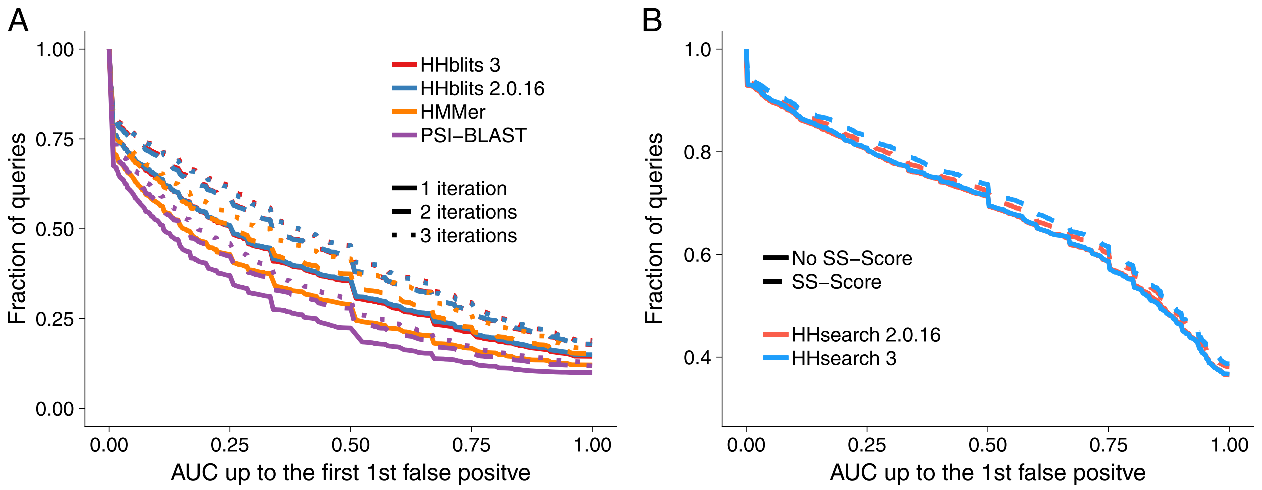


Fig. 7 Sensitivity of sequence search tools. **a** We searched with 6616 SCOP20 domain sequences through the UniProt plus SCOP20 database using one to three search iterations. The sensitivity to detect homologous sequences is measured by cumulative distribution of the Area Under the Curve 1 (AUC1), the fraction of true positives ranked better than the first false positive match. True positive matches are defined as being from the same SCOP superfamily [25], false positives have different SCOP folds, excepting known cases of inter-fold homologies. **b** Sensitivity of HHsearch with and without scoring secondary structure similarity, measured by the cumulative distribution of AUC1 for a comparison of 6616 profile HMMs built from SCOP20 domain sequences. Query HMMs include predicted secondary structure, target HMMs include actual secondary structure annotated by DSSP. True and false positives are defined as in A

through a database consisting of the uniprot20 HMMs plus the 6616 UniProt profile HMMs annotated by SCOP domains.

We defined a sequence match as true positive if query and matched sequence were from the same SCOP superfamily and as false positive if they were from different SCOP folds and ignore all others. We excluded the self-matches as well as matches between Rossman-like folds (c.2-c.5, c.27 and 28, c.30 and 31) and between the four- to eight-bladed β-propellers (b.66-b.70), because they are probably true homologs [2]. HMMER3 reported more than one false positive hit just in one out of three queries, despite setting the maximum E-value to 100 000, and we therefore measured the sensitivity up to the first false positive (AUC1) instead of the AUC5 we had used in earlier publications.

We ran HHblits using `hhblits -min_prefilter_hits 100 -n 1 -cpu $NCORES -ssm 0 -v 0 -wg` and wrote checkpoint files after each iteration to restart the next iteration. We ran HMMER3 (v3.1b2) using `hmmsearch -chkhmm -E 100000` and PSI-BLAST (2.2.31) using `-evalue 10000 -num_descriptions 250000`.

The cumulative distribution over the 6616 queries of the sensitivity at the first false positive (AUC1) in Fig. 7a shows that HHblits 3 is as sensitive as HHblits 2.0.16 for 1, 2, and 3 search iterations. Consistent with earlier results [5, 26], HHblits is considerably more sensitive than HMMER3 and PSI-BLAST.

We also compared the sensitivity of HHsearch 3 with and without scoring secondary structure similarity, because we slightly changed the weighting of the secondary structure score (Methods). We generated a profile HMM for each SCOP20 sequence using three search iterations with HHblits searches against the uniprot20 database of HMMs. We created the query set of profile HMMs by adding PSIPRED-based secondary structure predictions using the HH-suite script `addss.pl`, and we added structurally defined secondary structure states from DSSP [36] using `addss.pl` to the target profile HMMs. We then searched with all 6616 query HMMs through the database of 6616 target HMMs. True positive and false positive matches were defined as before.

Figure 7b shows that HHsearch 2.0.16 and 3 have the same sensitivity when secondary structure scoring is turned off. When turned on, HHsearch 3 has a slightly higher sensitivity due to the better weighting.

Conclusions

We have accelerated the algorithms most critical for runtime used in the HH-suite, most importantly the Viterbi algorithm for local and global alignments, using SIMD vector instructions. We have also added thread parallelization with OpenMP and parallelization across servers with Message Passing Interface (MPI). These extensions make the HH-suite well suited for large-scale deep protein annotation of metagenomics and genomics datasets.

Availability and requirements

- Project name: HH-suite
- Project page: https://github.com/soedinglab/hh-suite
- Operating systems: Linux, macOS
- Programming languages: C++, Python utilities
- Other requirements: support for SSE2 or higher
- License: GPLv3

Abbreviations

AVX2: advanced vector extension (SIMD instruction set standards); HMM: hidden Markov model; MSA: multiple sequence alignment; SIMD: single-instruction multiple-data; SSE2: streaming SIMD extensions 2

Acknowledgements

We thank the HH-suite community for their contributions and bug reports. We want to especially thank Lim Heo (Michigan State University) for fixing a bug in the Viterbi global alignment mode and David Miller for adding PowerPC support to the HH-suite.

Authors' contributions

MS & JS designed research, MS developed vectorized code and performed analyses, M. Meier refactored code, added features, fixed bugs and performed benchmarks, M. Mirdita added features, fixed bugs and maintains databases, HV implemented mmCIF support, SH optimized the MAC algorithm memory usage, MS and JS wrote the manuscript. All authors read and approved the final manuscript.

Funding

This work was supported by the European Research Council's Horizon 2020 Framework Programme for Research and Innovation ("Virus-X", project no. 685778).

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author on request.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Quantitative and Computational Biology Group, Max-Planck Institute for Biophysical Chemistry, Am Fassberg 11, 81379 Munich, Germany. [2]Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, MD, USA. [3]European Bioinformatics Institute, CB10 1SD Cambridge, United Kingdom. [4]Royal College of Surgeons, D02 YN77 Dublin, Ireland.

Received: 19 February 2019 Accepted: 2 August 2019
Published online: 14 September 2019

References

1. Howe AC, Jansson JK, Malfatti SA, Tringe SG, Tiedje JM, Brown CT. Tackling soil diversity with the assembly of large, complex metagenomes. Proc Natl Acad Sci USA. 2014;111(13):4904–4909. https://doi.org/10.1073/pnas.1402564111.
2. Söding J, Remmert M. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. Curr Opin Struct Biol. 2011;21(3):404–11. https://doi.org/10.1016/j.sbi.2011.03.005.
3. Eddy SR. A new generation of homology search tools based on probabilistic inference. Genome Inform. 2009;23(1):205–11.
4. Eddy SR. Accelerated Profile HMM Searches. PLOS Comput Biol. 2011;7(10):1002195. https://doi.org/10.1371/journal.pcbi.1002195.
5. Remmert M, Biegert A, Hauser A, Söding J. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat Methods. 2012;9(2):173–5. https://doi.org/10.1038/nmeth.1818.
6. Dill KA, MacCallum JL. The protein-folding problem, 50 years on. Science. 2012;338(6110):1042–6. https://doi.org/10.1126/science.121902.
7. Biasini M, Bienert S, Waterhouse A, Arnold K, Studer G, Schmidt T, Kiefer F, Cassarino TG, Bertoni M, Bordoli L, et al. SWISS-MODEL: modelling protein tertiary and quaternary structure using evolutionary information. Nucleic Acids Res. 2014;42(W1):252–8. https://doi.org/10.1093/nar/gku340.
8. Fidler DR, Murphy SE, Courtis K, Antonoudiou P, El-Tohamy R, Ient J, Levine TP. Using HHsearch to tackle proteins of unknown function: A pilot study with PH domains. Traffic. 2016;17(11):1214–26. https://doi.org/10.1111/tra.12432.
9. Burstein D, Harrington LB, Strutt SC, Probst AJ, Anantharaman K, Thomas BC, Doudna JA, Banfield JF. New CRISPR-Cas systems from uncultivated microbes. Nature. 2016;542:237. https://doi.org/10.1038/nature21059.
10. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25(17):3389–402. https://doi.org/10.1093/nar/25.17.3389.
11. Rychlewski L, Jaroszewski L, Li W, Godzik A. Comparison of sequence profiles. Strategies for structural predictions using sequence information. Protein Sci. 2000;9(2):232–41. https://doi.org/10.1110/ps.9.2.232.
12. Sadreyev R, Grishin N. COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. J Mol Biol. 2003;326(1):317–36. https://doi.org/10.1016/S0022-2836(02)01371-2.
13. Zhang W, Liu S, Zhou Y. SP5: Improving Protein Fold Recognition by Using Torsion Angle Profiles and Profile-Based Gap Penalty Model. PloS One. 2008;3(6):2325. https://doi.org/10.1371/journal.pone.0002325.
14. Margelevičius M, Venclovas Č. Detection of distant evolutionary relationships between protein families using theory of sequence profile-profile comparison. BMC Bioinform. 2010;11(1):89. https://doi.org/10.1186/1471-2105-11-89.
15. Söding J. Protein homology detection by HMM-HMM comparison. Bioinformatics. 2005;21(7):951–60. https://doi.org/10.1093/bioinformatics/bti125.
16. Edgar RC. Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010;26(19):2460–1. https://doi.org/10.1093/bioinformatics/btq461.
17. Kielbasa SM, Wan R, Sato K, Horton P, Frith M. Adaptive seeds tame genomic sequence comparison. Genome Res. 2011;21(3):487–93. https://doi.org/10.1101/gr.113985.110.
18. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. Nat Methods. 2014;12(1):59–60. https://doi.org/10.1038/nmeth.3176.
19. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol. 2017;35(11):1026–8. https://doi.org/10.1038/nbt.3988.
20. El-Gebali S, Mistry J, Bateman A, Eddy SR, Luciani A, Potter SC, Qureshi M, Richardson LJ, Salazar GA, Smart A, et al. The Pfam protein families database in 2019. Nucleic Acids Res. 2018;47(D1):427–32. https://doi.org/10.1093/nar/gky995.
21. Mitchell AL, Attwood TK, Babbitt PC, Blum M, Bork P, Bridge A, Brown SD, Chang H.-Y., El-Gebali S, Fraser MI, et al. Interpro in 2019: improving coverage, classification and access to protein sequence annotations. Nucleic Acids Res. 2018;47(D1):351–60.
22. Biegert A, Söding J. De novo identification of highly diverged protein repeats by probabilistic consistency. Bioinformatics. 2008;24(6):807–14. https://doi.org/10.1093/bioinformatics/btn039.
23. Mirdita M, von den Driesch L, Galiez C, Martin MJ, Söding J, Steinegger M. Uniclust databases of clustered and deeply annotated protein sequences and alignments. Nucleic Acids Res. 2016;45(D1):170–6. https://doi.org/10.1093/nar/gkw1081.
24. Gilliland G, Berman HM, Weissig H, Shindyalov IN, Westbrook J, Bourne PE, Bhat TN, Feng Z. The Protein Data Bank. Nucleic Acids Res. 2000;28(1):235–42. https://doi.org/10.1093/nar/28.1.235.
25. Andreeva A, Howorth D, Chandonia J-M, Brenner SE, Hubbard TJ, Chothia C, Murzin AG. Data growth and its impact on the SCOP database:

new developments. Nucleic Acids Res. 2007;36(Database issue):419–25. https://doi.org/10.1093/nar/gkm993.

26. Angermüller C, Biegert A, Söding J. Discriminative modelling of context-specific amino acid substitution probabilities. Bioinformatics. 2012;28(24):3240–7. https://doi.org/10.1093/bioinformatics/bts622.
27. Eddy SR. Profile hidden Markov models. Bioinformatics. 1998;14(9): 755–63. https://doi.org/10.1093/bioinformatics/14.9.755.
28. Li ITS, Shum W, Truong K. 160-fold acceleration of the Smith-Waterman algorithm using a field programmable gate array (FPGA). BMC Bioinform. 2007;8(1):185. https://doi.org/10.1186/1471-2105-8-185.
29. Manavski SA, Valle G. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. BMC Bioinform. 2008;9 Suppl 2(Suppl 2):10. https://doi.org/10.1186/1471-2105-9-S2-S10.
30. Szalkowski A, Ledergerber C, Krähenbühl P, Dessimoz C. SWPS3 - fast multi-threaded vectorized Smith-Waterman for IBM Cell/B.E. and x86/SSE2. BMC Res Notes. 2008;1(1):107. https://doi.org/10.1186/1756-0500-1-107.
31. Liu Y, Maskell DL, Schmidt B. CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units. BMC Res Notes. 2009;2(1):73. https://doi.org/10.1186/1756-0500-2-73.
32. Wozniak A. Using video-oriented instructions to speed up sequence comparison. Bioinformatics. 1997;13(2):145–50. https://doi.org/10.1093/bioinformatics/13.2.145.
33. Rognes T, Seeberg E. Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. Bioinformatics. 2000;16(8):699–706. https://doi.org/10.1093/bioinformatics/16.8.699.
34. Farrar M. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. Bioinformatics. 2007;23(2):156–61. https://doi.org/10.1093/bioinformatics/btl582.
35. Rognes T. Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation. BMC Bioinform. 2011;12(1):221. https://doi.org/10.1186/1471-2105-12-221.
36. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983;22(12):2577–637. https://doi.org/10.1002/bip.360221211.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

5.3 MetaEuk—sensitive, high-throughput gene discovery, and annotation for large-scale eukaryotic metagenomics

Publication:

MetaEuk—sensitive, high-throughput gene discovery, and annotation for large-scale eukaryotic metagenomics

E. Levy Karin†, **M. Mirdita**, J. Söding†

(†) corresponding author

Microbiome (2020), 8, 48.
Cited 31 times since 12/2020.

Code and software availability

MetaEuk is available as free open source software (GPLv3) at `metaeuk.soedinglab.org`.

Author contributions

E.L.K. & J.S. designed the MetaEuk algorithm, benchmark, and biological application. E.L.K. & **M.M.** developed the algorithm. E.L.K. analyzed the benchmark and Tara Oceans data. E.L.K. & **M.M.** generated the figures. E.L.K., J.S., & **M.M.** drafted the manuscript.

Microbiome

RESEARCH Open Access

MetaEuk—sensitive, high-throughput gene discovery, and annotation for large-scale eukaryotic metagenomics

Eli Levy Karin*, Milot Mirdita and Johannes Söding*

Abstract

Background: Metagenomics is revolutionizing the study of microorganisms and their involvement in biological, biomedical, and geochemical processes, allowing us to investigate by direct sequencing a tremendous diversity of organisms without the need for prior cultivation. Unicellular eukaryotes play essential roles in most microbial communities as chief predators, decomposers, phototrophs, bacterial hosts, symbionts, and parasites to plants and animals. Investigating their roles is therefore of great interest to ecology, biotechnology, human health, and evolution. However, the generally lower sequencing coverage, their more complex gene and genome architectures, and a lack of eukaryote-specific experimental and computational procedures have kept them on the sidelines of metagenomics.

Results: MetaEuk is a toolkit for high-throughput, reference-based discovery, and annotation of protein-coding genes in eukaryotic metagenomic contigs. It performs fast searches with 6-frame-translated fragments covering all possible exons and optimally combines matches into multi-exon proteins. We used a benchmark of seven diverse, annotated genomes to show that MetaEuk is highly sensitive even under conditions of low sequence similarity to the reference database. To demonstrate MetaEuk's power to discover novel eukaryotic proteins in large-scale metagenomic data, we assembled contigs from 912 samples of the Tara Oceans project. MetaEuk predicted >12,000,000 protein-coding genes in 8 days on ten 16-core servers. Most of the discovered proteins are highly diverged from known proteins and originate from very sparsely sampled eukaryotic supergroups.

Conclusion: The open-source (GPLv3) MetaEuk software (https://github.com/soedinglab/metaeuk) enables large-scale eukaryotic metagenomics through reference-based, sensitive taxonomic and functional annotation.

Keywords: MetaEuk, Eukaryotes, Homology detection, Prediction, Annotation, Contigs, Marine

Background

Unicellular eukaryotes are present in almost all environments, including soil [1], oceans [2], and plant and animal-associated microbiomes [3, 4]. They exhibit both autotrophic and heterotrophic lifestyles [5], exist in symbiosis with plants and animals [6], and interact with other microbial organisms [7]. They account for roughly half of the global primary productivity in the oceans, mostly by photosynthesis [8], are key contributors to the carbon and nitrogen cycles through carbon-dioxide fixation, organic matter degradation, and denitrification [9, 10], and have been shown to be a source for chemically bioactive compounds [11, 12].

Since the advent of metabarcoding using 18S rRNA genes, the known evolutionary diversity of unicellular eukaryotes has increased by orders of magnitude [13], and novel phyla and supra-kingdoms are still being discovered [14, 15]. Due to their vast diversity [16, 17], unicellular

* Correspondence: eli.levy.karin@gmail.com; soeding@mpibpc.mpg.de
Quantitative and Computational Biology, Max-Planck Institute for Biophysical Chemistry, 37077 Göttingen, Germany

eukaryotes are certain to hold invaluable secrets for biotechnology and biomedicine.

Protein-coding genes are major keys for understanding eukaryotic functions and activities [18]. Metatranscriptomic and metagenomic studies provide unique means to reveal protein-coding genes. However, despite the great potential of studying uncultivatable eukaryotes in their natural environment, they have received little attention in metatranscriptomic and metagenomic studies so far, with a few notable exceptions e.g., [19, 20]. The unique features of eukaryotic data, i.e., lower genomic coverage due to lower population densities in metagenomic samples, fewer reference genomes, increased genome sizes, and higher complexity of gene structure negatively impact all stages of metagenomic analyses, from assembly, through binning, to protein prediction and annotation [21, 22].

Specifically, identifying protein-coding genes in eukaryotes is inherently more challenging than in prokaryotes due to the exon-intron architecture of eukaryotic genes. To date, methods for eukaryotic gene calling e.g., [23–25] consider two types of information when training models for gene prediction: intrinsic sequence signals (e.g., CpG islands) and extrinsic data, such as transcriptomics or an annotated genome from a closely related organism. As splicing signatures are not well conserved throughout evolution, the predictive power of the trained models declines fast when applied to organisms that are phylogenetically distant from the organism on which the model was trained [26].

While these methods are very useful for genomics, their applicability to metagenomic data is severely limited. First, the transcriptomic or genomic data of annotated organisms that are sufficiently closely related are usually not available. Second, since the models need to be trained on a relatively narrow clade, the application of such methods to metagenomic data requires to first bin the assembled contigs by their assumed genome of origin as performed by [27], which is often quite inaccurate and slow, especially when the number of contigs is large, the coverage is low, the contigs are short, and the metagenomic data are species-rich [28–30]. Finally, model-training in itself is time consuming, taking hours to days per genomic bin [25, 27], limiting this approach to the analysis of few genomic bins at a time.

Previously, methods that bypass or reduce the need to explicitly train models to detect protein-coding genes have been proposed in the context of genomics e.g., [31, 32]. These methods extract putative protein-coding fragments from the genome and join those that bear sequence similarity to available transcriptomic or protein sequence targets. Since the joined fragments can be separated by non-coding (intronic) regions, their match to the target is termed “spliced alignment.” Even at a genomic level, a brute force application of the spliced alignment approach poses a serious computational burden as it requires aligning each putative fragment to each target as well as recovering the set of putative fragments that best match a target.

Here, we developed MetaEuk, a novel and sensitive reference-based approach to identify single- and multi-exon protein-coding genes in eukaryotic metagenomic data. MetaEuk takes as input a set of assembled contigs and a reference database of target protein sequences or profiles. MetaEuk scans each contig in all six reading frames and extracts putative protein fragments between stop codons in each frame. Thus, MetaEuk makes no assumption about the splicing signal and does not rely on any preceding binning step. MetaEuk uses the MMseqs2 code library [33] for a very fast, yet sensitive identification of putative exons within the fragments. This step also discards the vast majority of fragments, which significantly reduces the computation time of all succeeding steps. The combinatorial task of considering all possible sets of putative exons to best match a given target is solved by means of dynamic programming. Since MetaEuk uses a homology-based strategy to identify protein-coding genes, it can directly confer annotations to the discovered genes from the matched target proteins.

We benchmarked MetaEuk by using annotated genomes and proteins of seven unicellular organisms from different parts of the eukaryotic tree of life under conditions of increasing evolutionary distance to sequences in the reference database. Despite its high speed and low false positive rates, MetaEuk is able to discover a large fraction of the known proteins in these benchmark genomes. We next applied MetaEuk to study marine eukaryotes. We assembled all Tara Oceans metagenomic samples [20] and focused on ~1,300,000 contigs of at least 5 kbp in length. We clustered more than 330,000,000 proteins to create a comprehensive catalog of over 87,000,000 protein profiles to serve as a reference database. We found the MetaEuk collection of >12,000,000 marine proteins is highly diverged, offering major eukaryotic lineage expansions.

Results

The MetaEuk algorithm

The main steps of the algorithm are presented schematically in Fig. 1, and a detailed description is provided in the Methods section. For each input contig, all possible protein-coding fragments are translated in six reading frames and searched against a reference target database of protein sequences or profiles. Fragments from the same contig and strand that hit a reference target *T* are examined together. In each fragment, only the part that was aligned to the target protein *T* is considered as a putative exon. The putative exons are ordered according to their start position on the

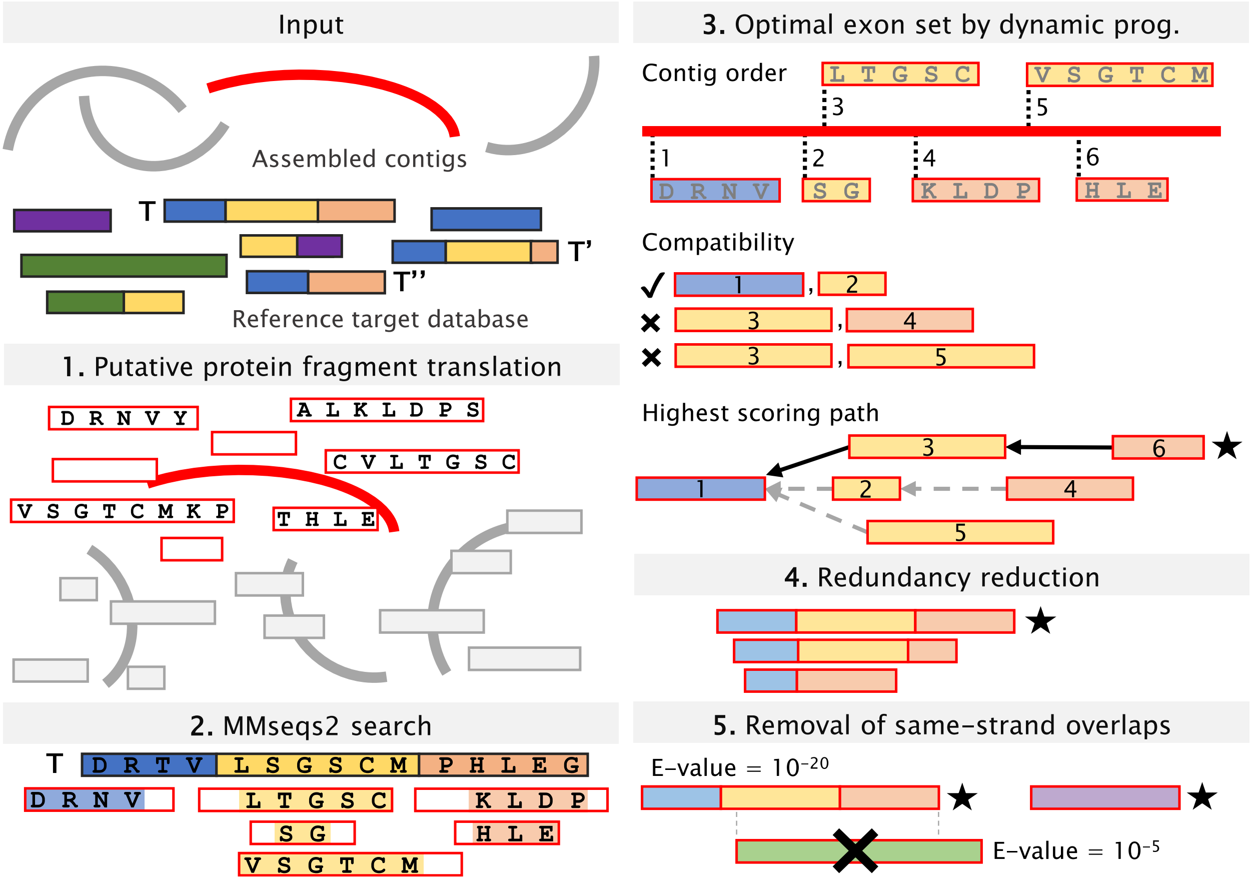


Fig. 1 MetaEuk algorithm. Input to MetaEuk are assembled metagenomic contigs and a reference database of protein sequences. **1** Six-frame translation of all putative protein-coding fragments from each contig. **2** Fragments on the same contig and strand that hit the same reference protein *T* are examined together. **3** Putative exons are identified and ordered according to their start position on the contig. The highest score and path (denoted with a star) of a set of compatible putative exons is computed by dynamic programming, in which individual scores of the putative exons are summed and unmatched amino acids are penalized. **4** Redundancies among gene calls due to homologous targets (*T*, *T'*, and *"T'*) are reduced and a representative prediction (denoted with a star) is retained. **5** Contradicting predictions of overlapping genes on the same strand are resolved by excluding the prediction with the higher *E*-value

contig. Based on their contig locations and the locations of their aligned region on the target *T*, any two putative exons are either compatible or not. A dynamic programming procedure recovers the highest scoring path of compatible pairs of putative exons by computing the maximum scores of all paths ending with each putative exon. Since homologies among targets in the reference database can lead to multiple calls of the same protein-coding gene, redundancies are reduced by clustering the calls. To that end, all calls are ordered by their start position on the contig. The first call defines a new cluster and all calls that overlap it on the contig are assigned to its cluster if they share an exon with it. The next cluster is defined by the first unassigned call. After all calls are clustered, the best scoring call is selected as the representative of the cluster, termed a "prediction." Finally, as overlaps of genes on the same strand are very rare as reviewed by [34], gene predictions overlapping others on the same strand with a better *E*-value are removed.

Performance evaluation on benchmark data

We evaluated MetaEuk using seven annotated unicellular eukaryotic organisms obtained from the NCBI's genome assembly database [35] (Table 1). These organisms are varied in terms of their phylogenetic group, genome size, number of annotated proteins, fraction of multi-exon genes, and assembly quality. MetaEuk was run on the assembled scaffolds of each of these organisms against the UniRef90 [36] database with an average run time of 42 min per genome, or 0.5 Mbp/min, on a server with two 8-core Intel Xeon E5-2640v3 CPUs and 128 GB RAM (Table 1). The NCBI data included the scaffold coordinates of the annotated protein-coding genes and their exons. In the following sections, we used this information to assess MetaEuk's sensitivity and precision by mapping MetaEuk predictions to annotated proteins in their scaffold location. This was done based on the scaffold boundaries of the MetaEuk prediction and the

Table 1 Species used to benchmark MetaEuk

Species	Group	Genome size (Mbp)	# scaffolds	# annotated proteins	% multi-exon proteins	GC%	MetaEuk run time against UniRef90
Schizosaccharomyces pombe	Fungi	12.59	4	5132	47	36	35 m
Acanthamoeba castellanii str. Neff	Amoebozoa	42.02	384	14,974	91	57.8	59 m
Phytomonas sp. isolate EM1	Excavata	17.78	138	6381	0	48	37 m
Babesia bigemina	Alveolates	13.84	483	5079	54	50.6	35 m
Nucleomorph of *Lotharella oceanica*	Rhizaria	0.68	4	668	39	32.8	24 m
Phaeodactylum tricornutum	Stramenopiles	27.45	88	10,408	46	48.8	51 m
Aspergillus nidulans	Eurotiomycetes	30.28	91	9556	88	50.3	52 m

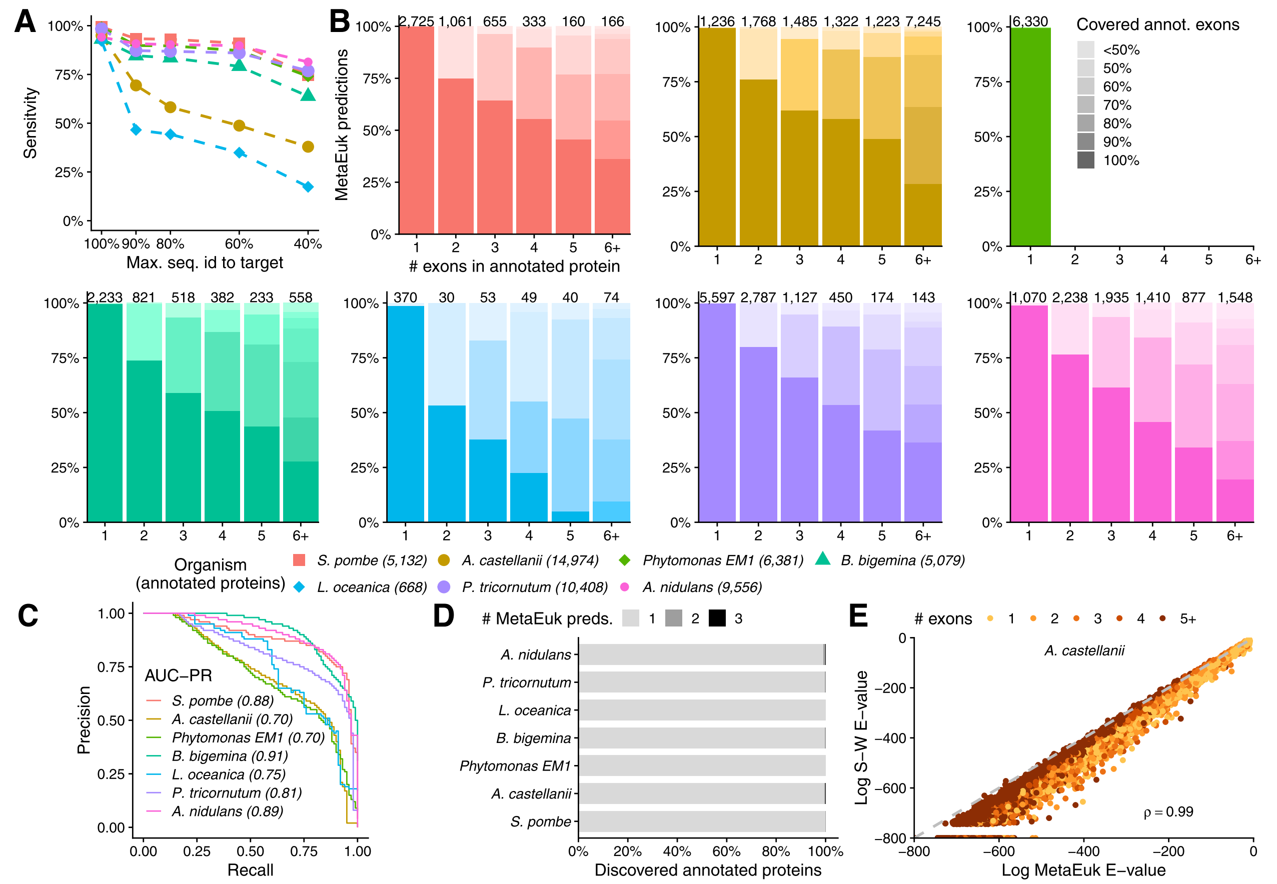


Fig. 2 MetaEuk evaluation on benchmark. MetaEuk predictions were mapped to annotated proteins. **a** Conditions of increasing evolutionary divergence were simulated by excluding gene calls based on their sequence identity to their target. Sensitivity is the fraction of annotated proteins from the query genome to which a MetaEuk prediction was mapped. **b** Fraction of exons covered by MetaEuk (color saturation). The number of MetaEuk predictions is indicated on top of each bar. **c** In an annotation-dependent precision estimation, MetaEuk predictions that mapped to an annotated protein were considered as "true" and the rest as "false." These sets of predictions are well separated by their *E*-values, as indicated by the high AUC-PR values. **d** Fraction of annotated protein-coding genes that were split by MetaEuk into two (dark grey) or three (black) different predictions. **e** Comparison of the *E*-values computed by MetaEuk and by the Smith-Waterman algorithm for *A. castellani* proteins. The Spearman rho values indicate high correlation for *A. castellani* and for the other organisms (Supp. Figure S3A)

annotated protein and by requiring high sequence identity of their protein alignment. We then computed the coverage of individual exons of the annotated proteins to which MetaEuk predictions were mapped. These mappings are fully described in the Methods section.

Sensitivity at evolutionary distance

Sequences from major eukaryotic clades, such as Rhizaria, Stramenopiles, and Dinoflagellata are poorly represented in public protein databases, despite their high abundance in the environment [17]. We therefore measured the ability of MetaEuk to identify homologous protein-coding genes in organisms, which have distant evolutionary relatives in the reference database, as would be the case in a typical metagenomic analysis. To that end, for each annotated organism, we considered five sets of MetaEuk predictions. The first is the base set, which consisted of all predictions. Since we worked with annotated species, their proteins are well represented in UniRef90. The base set therefore reflects ideal conditions, in which the queried organisms are close to the reference database. The other four sets reflect an increasing evolutionary distance and were generated by excluding MetaEuk gene calls whose Smith-Waterman alignment (computed using MMseqs2) to their UniRef90 target had more than 90%, 80%, 60%, or 40% sequence identity. We measured sensitivity as the fraction of annotated proteins from the query genome to which a MetaEuk prediction was mapped (see Methods). For all organisms, the sensitivity of the base set of predictions was at least 92%, and sensitivity decreased with the sequence identity threshold (Fig. 2a). However, even at low thresholds (40–60%), a significant fraction of the annotated proteins could be discovered.

Annotated exon coverage

We next assessed MetaEuk's performance at the level of individual exons. For each MetaEuk prediction from the base set and its mapped annotated protein, we computed the proportion of annotated exons that were covered by the prediction (see Methods). Overall, the majority of predictions covered the majority of exons and, as expected, the fraction of predictions that cover all annotated exons decreases with the number of exons in the annotated protein (Fig. 2b). For all organisms, most (77–91%) annotated exons were covered by MetaEuk predictions. In addition, we found that the fraction of multi-exon MetaEuk predictions was similar to that presented in Table 1 (average difference 10%, Supp. Figure S1A) and that single-exon predictions tended to have longer exons than multi-exon predictions (Supp. Figure S1B). An additional measure of completeness of MetaEuk predictions is the coverage of the target UniRef90 protein based on which the prediction was made. We therefore aligned each predicted MetaEuk protein to its target and found that on average, >83% of predictions covered >90% of their target (Supp. Figure S2).

Precision

MetaEuk predictions that were mapped to annotated proteins were considered as true predictions. We first measured the precision of MetaEuk by using the NCBI annotations as gold standard and regarded all predictions in the base set that were not mapped to an annotated protein (8–35%, Supp. Figure S2) as false. We computed precision-recall curves by treating the predictions' *E*-values as a classifying score. We found good separation (AUC-PR > 0.7 in all cases) between predictions that mapped to annotated proteins and the rest (Fig. 2c). However, a prediction that does not map to a known protein is not necessarily false as it might reflect an unannotated protein. We found that about 40% of the unmapped predictions overlap a protein-coding gene on the opposite strand or are on scaffolds that had no annotation at all (Supp. Figure S2), suggestive of post hoc exclusion criteria in the NCBI annotation procedure. For this reason, we also measured the precision of MetaEuk independently of external annotations by using an inverted sequence null model. For this annotation-free approach, we ran standard MetaEuk on the inverted sequences of the six frame-translated putative fragments. Each prediction based on these inverted sequences can therefore be considered a false positive. We applied the same *E*-value cutoff for reporting predictions based on the original sequence data and based on the inverted set. For all organisms, the total number of false positive predictions produced by this approach was low (0–12), indicating very high precision (> 99.9%).

Redundancy reduction

MetaEuk's redundancy reduction procedure divides gene calls into disjoint clusters and retains a representative call as gene prediction for each cluster (see Methods). This reduces the number of potential protein-coding genes that need to be inspected. For example, for *S. pombe*, MetaEuk produced over 1,100,000 calls that were reduced to a total of 5564 predictions in the base set. A full reduction of redundancy is achieved when no two predictions correspond to same protein-coding gene. We thus identified cases in which two or more MetaEuk predictions were mapped to the same protein-coding gene. We found that for all benchmark organisms, redundancy is greatly reduced, as more than 99% of the annotated protein-coding genes in the benchmark scaffolds are only predicted once (Fig. 2d).

Statistical scores

For each prediction, MetaEuk computes a bit-score between the set of translated and joined putative exons and the target protein. Based on this bit-score and the size of the reference database, an *E*-value is computed (see Methods). We evaluated MetaEuk's bit-scores and *E*-values by comparing them to those computed for each predicted protein and its target by the Smith-Waterman algorithm. Since MetaEuk penalizes missing and overlapping amino acids when joining putative exons, we expect the MetaEuk bit-score to be more conservative than the direct Smith-Waterman alignment bit-score. We found very high levels of agreement between the MetaEuk statistics and the Smith-Waterman statistics (Fig. 2e, Supp. Figure S3). This suggests a straightforward statistical interpretation of MetaEuk prediction scores.

Effect of contig length

Assembling metagenomic reads often produces contigs that are much shorter than the scaffolds of the organisms we used for benchmarking MetaEuk (Table 1). We thus aimed to assess the effect of analyzing shorter genomic stretches by artificially dividing each of the scaffolds from Table 1 into shorter contigs following a typical length distribution with a minimum of 5 kbp in length and a median of 6.8 kbp (see Methods). Any protein-coding gene that spans more than one contig is expected to result in incomplete MetaEuk predictions. Indeed, while the sensitivity measured by the mapping to annotated proteins remained similar to that recorded on the original scaffolds (Supp. Figure S4A), we found that more predictions were partial and covered fewer annotated exons (Supp. Figure S4B) as well as an increase of up to 15% in annotated genes being split into more than one MetaEuk prediction (Supp. Figure S4D).

Eukaryotic protein-coding genes in the ocean

To date, little is known about the biological activities of eukaryotes in the oceans [2, 37]. We aimed to use MetaEuk to discover eukaryotic protein-coding genes in the Tara Oceans metagenomic dataset [20]. We first used MEGAHIT [38] to assemble all 912 samples of this project. We retained 1,351,204 contigs of at least 5 kbp in length that were classified as potentially eukaryotic by EukRep [27]. We next constructed a comprehensive set of reference proteins by uniting over 21,000,000 representative sequences of the Uniclust50 database [39], the MERC dataset of over 292,000,000 protein sequence fragments assembled from eukaryotic Tara Oceans metatranscriptomic datasets [40], and over 18,500,000 protein sequences of MMETSP, the Marine Microbial Eukaryotic Transcriptome Sequencing Project [17, 41]. We clustered the joint dataset of 331,913,793 proteins using the combined Linclust/MMseqs2 four-step cascaded clustering workflow [42] with a minimal sequence identity of 20% and high sensitivity (-s 7). This resulted in 87,984,812 clusters, most of which (> 97%) contained proteins from a single reference dataset (Fig. 3). For each cluster, a multiple sequence alignment was generated, based on which a sequence profile was computed.

MetaEuk's run using this reference database took 8 days on ten 2x8-core servers and resulted in 12,111,301 predictions with no same-strand overlaps in 1,287,197 of the Tara Oceans contigs. Due to sequence similarities among the assembled contigs, some of these proteins are identical to each other, leaving a total of 6,158,526 unique proteins. We examined the distribution of predictions per contig, the number of putative exons in each prediction and the length of putative exons in single-exon and multi-exon predictions. We found that the number of predictions increases as a function of the contig length (Fig. 4a), about 24% of predictions had more than one putative exon (Fig. 4b) and multi-exon predictions tend to have shorter putative exons than single-exon predictions (Fig. 4c). We analyzed the contribution of each reference dataset to the profiles based on which the MetaEuk predictions were made. MERC, MMETSP, and Uniclust50 contributed 77.4%, 5.7%, and 4.3% of the predictions, respectively. The rest of the predictions were based on mixed-dataset clusters (Supp.

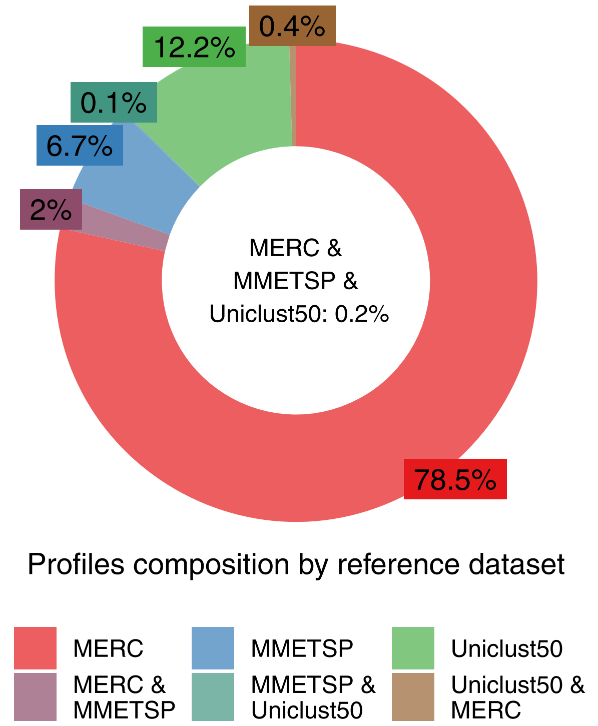


Fig. 3 Reference profiles composition. Proteins from three datasets: MERC (292 million), MMETSP (18.5 million), and Uniclust50 (21 million) were clustered into ~ 88 million clusters. Most clusters contained proteins from a single reference dataset. The profiles computed based on these clusters served as the reference database for the MetaEuk run on the Tara Oceans contigs

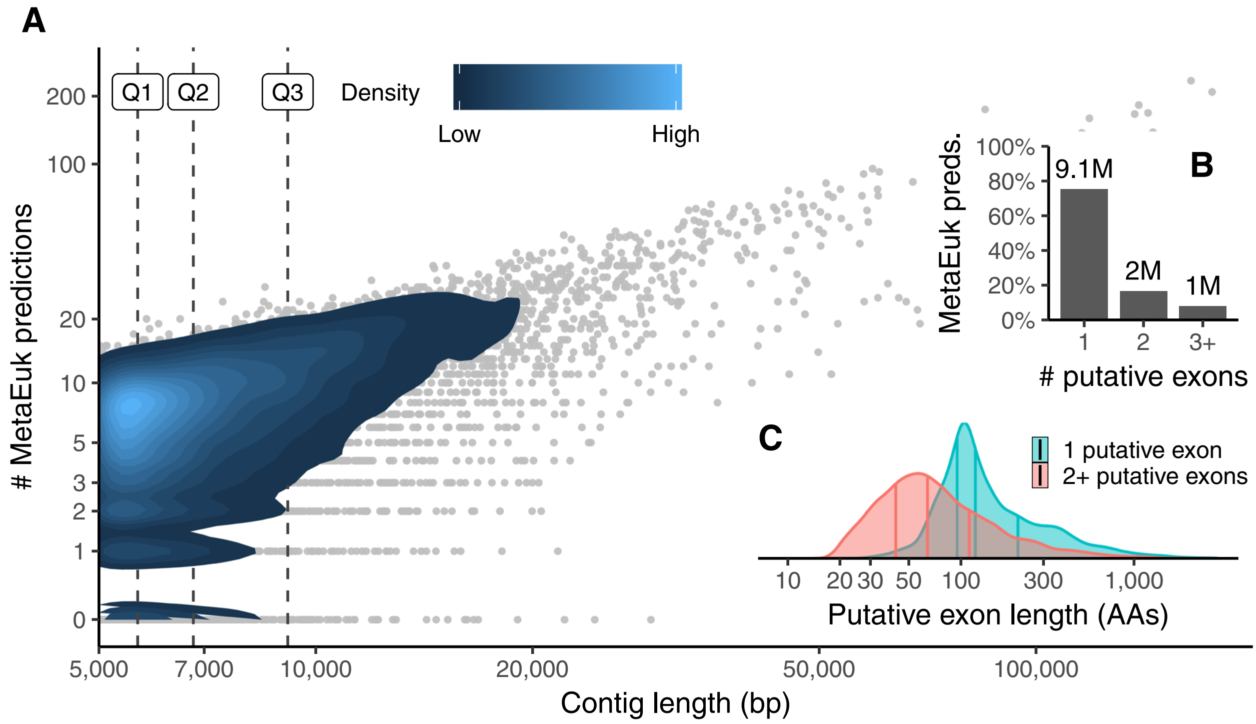


Fig. 4 MetaEuk predictions on Tara Oceans contigs. MetaEuk was run on over 1.3 million contigs assembled from Tara Oceans metagenomic reads against a reference database of ~88 million protein profiles. **a** The number of MetaEuk predictions per contig increases with its length. Horizontal lines mark contig length quartiles. **b** Most (76%) MetaEuk predictions had a single putative exon. The absolute number of predictions is indicated above each bar. **c** Single-exon predictions tend to have longer putative exons than multi-exon predictions

Figure S5). We then used MMseqs2 to query the MetaEuk predicted proteins against their targets. Over 33% of the MetaEuk predictions have less than 60% sequence identity to their MERC, MMETSP, or Uniclust50 target (Fig. 5a). Finally, we found that 70% of the MetaEuk predicted proteins covered at least 80% of their reference target (Fig. 5b).

We next explored the taxonomic composition of the MetaEuk proteins. Since the majority (77%) of MetaEuk predictions were based on homologies to the MERC dataset, for which no taxonomic annotation is available, we queried the MetaEuk marine protein collection against the Uniclust90 dataset [39] and the MMETSP dataset, both annotated using NCBI taxonomy (see Methods). We found that 63% of predictions based on homologies to the MERC dataset did not match any protein in either of the reference datasets, which means ~49% (63% of 77%) of the MetaEuk marine protein collection could not be assigned any taxonomy. This is in agreement with 52% of unassigned unigenes assembled from Tara Oceans metatranscriptomics [20]. We next assigned taxonomic labels to each assembled contig by conferring the taxonomic label with the best *E*-value of all MetaEuk predictions in the contig. This allowed us to annotate 92% of the contigs for which MetaEuk produced predictions (87% of all input contigs). We found that 82% of the contigs were assigned to the domain Eukaryota and 9% to non-eukaryotes, mostly bacteria (Fig. 6a). We then examined the assigned eukaryotic supergroups below the domain level. About 12% of the eukaryotic contigs could not be assigned a supergroup. Among the most abundant eukaryotic supergroups are Metazoa and Chlorophyta (Fig. 6b).

The high fraction of unassignable predictions (49%) prompted us to seek an additional way to assess the diversity of the MetaEuk marine proteins. We thus collected orthologous sequences of the large subunits of RNA polymerases, which are universal phylogenetic markers [43] from 985 organisms for which we had taxonomic information, as well as 1076 MetaEuk proteins, which consisted of all five Pfam domains of the large subunit in the right order (see Methods). We aligned these sequences using MAFFT [44] and constructed the maximum-likelihood phylogeny using RAxML [45]. The aim of this analysis was to delineate the diversity of eukaryotic taxa of the MetaEuk marine protein collection and not to resolve the exact phylogenetic relationships among them. As can be seen in Fig. 7, MetaEuk proteins offer major lineage expansions in under-sampled eukaryotic supergroups. Importantly, the strict ortholog collection procedure performed for this analysis results in a conservative estimate of the diversity level of the MetaEuk marine proteins collection.

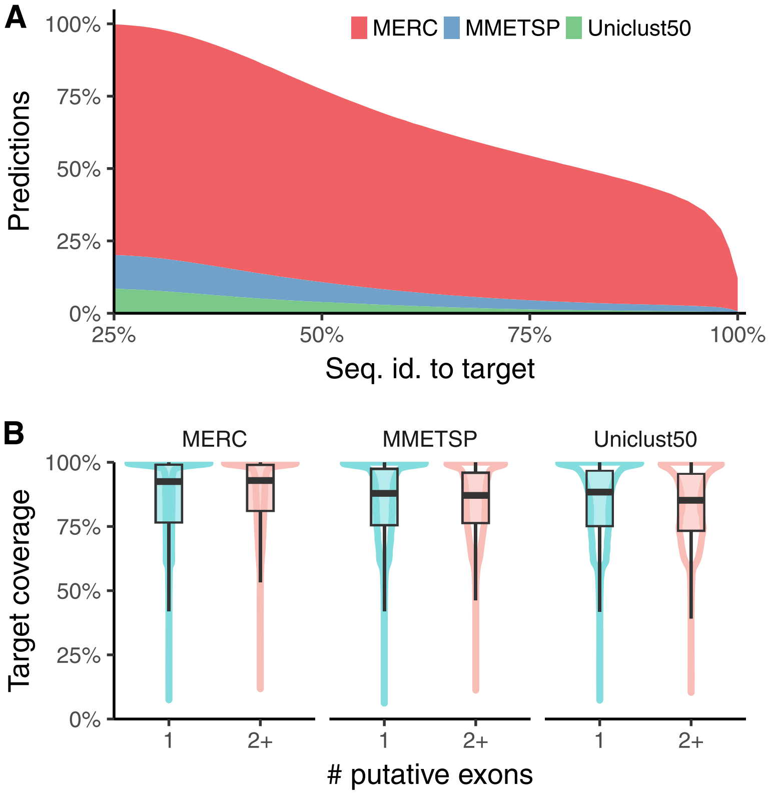


Fig. 5 MetaEuk predictions compared to the reference datasets. MetaEuk predicted proteins were queried against the representative sequence of their target reference cluster. **a** About one third of the predicted MetaEuk proteins had less than 60% sequence identity to their target. **b** Targets are well covered by MetaEuk predicted proteins

Discussion

We presented MetaEuk, an algorithm designed for large-scale analysis of eukaryotic metagenomic data. We demonstrated its utility for discovering proteins from highly diverged eukaryotic groups by analyzing assemblies of a huge set of 912 marine metagenomics samples. MetaEuk makes no assumption concerning splice site signatures and does not require a preceding binning procedure, which renders it suitable for the analysis of contigs from a mixture of highly diverged organisms. In order to achieve this, MetaEuk considers all possible putative protein-coding fragments from each input contig. Applying the spliced alignment dynamic programming procedure to recover the optimal set of putative exons directly on these fragments would result in a run time complexity per contig that is quadratic in the number of its fragments times the number of targets in the reference database. This is not feasible for metagenomics, as the number of fragments can be very high (e.g., from 1,351,204 Tara Oceans contigs, 152,519,258 fragments were extracted) and the reference database should be as comprehensive as possible (in this study, we used more than 87,000,000 protein profiles). To circumvent this limitation, MetaEuk takes advantage of the ultra-fast MMseqs2 search algorithm, which allows it to find putative exons matching a reference protein sequence with sufficient significance (in this study, a lenient *E*-value of 100). MetaEuk does not require significance at the exon level as it can combine sub-significant single exon matches to highly significant multi-exon matches. For example, two putative exons each with an *E*-value of 10 (corresponding to a bit-score of 25–40 in this study), are not individually significant but the sum of their bit-scores of at least 50 corresponds to a significant *E*-value of 10^{-5}.

MetaEuk is not designed to recover accurate splice sites, but rather to identify the protein-coding parts within exons. Indeed, we showed that MetaEuk predictions on the benchmark covered the majority (77–91%) of exons in annotated proteins. Since MetaEuk relies on local alignment at the amino acid level, it could potentially report pseudogenes, which still bear sequence similarity to reference proteins. However, we found that the majority of benchmark predictions (65–92%) mapped to NCBI annotated protein-coding genes, while the rest could be well separated from those that mapped by their *E*-values (AUC-PR > 0.7). Furthermore, unmapped predictions can reflect a missing annotation or post hoc

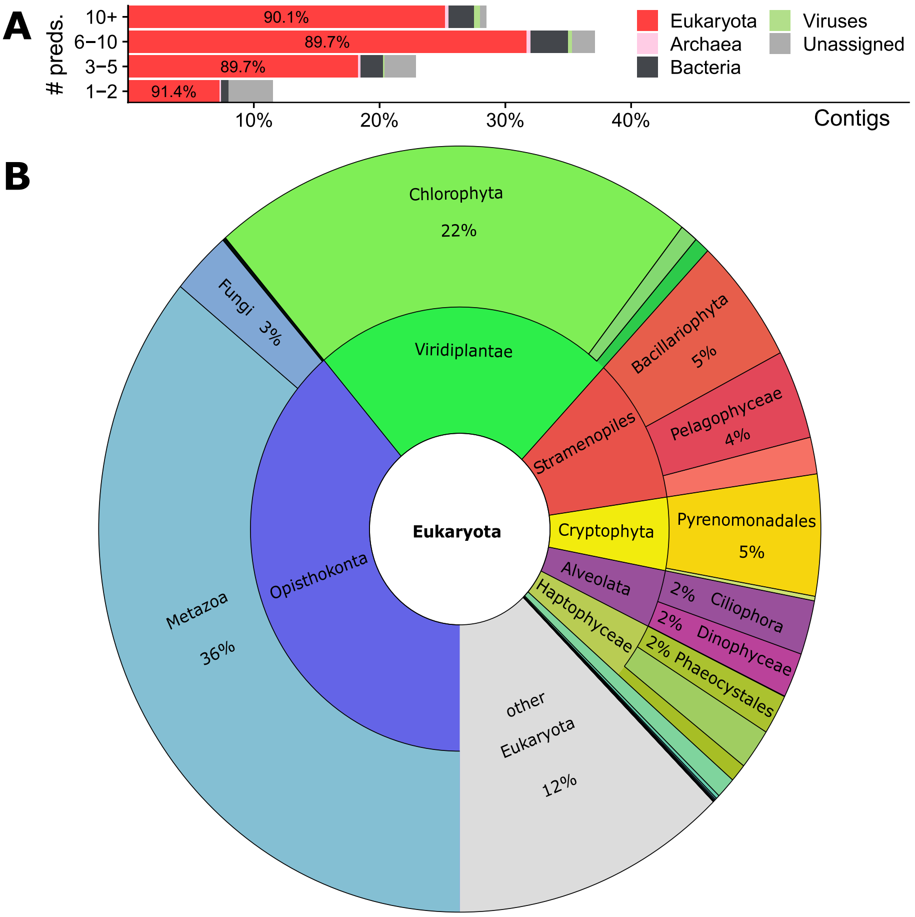


Fig. 6 Taxonomy of Tara Oceans contigs with MetaEuk predictions. The best-scoring taxonomic label of all predictions on each contig was conferred to the contig. Contigs were divided into four categories according to their number of MetaEuk predictions. Over 82% of the contigs were assigned to the domain Eukaryota. **a** The proportion of unassigned contigs decreases with the number of MetaEuk predictions on the contig. The fraction of eukaryotic contigs out of all assigned contigs is about 90% in all four categories. **b** Eukaryotic taxonomic labels below the domain level

exclusion criteria (e.g., removal of annotations that overlap a better scoring one on the opposite strand). We therefore measured precision independently of annotations by running standard MetaEuk on the inverted sequences of the putative protein fragments extracted from the contigs. By using this annotation-free approach, we showed that MetaEuk's precision was greater than 99.9% for all benchmark organisms. Put together, MetaEuk's strength is in describing the protein-coding repertoire of versatile environments rather than in constructing statistical models of exon-intron transitions.

The Tara Oceans contigs analyzed in this study were assembled from Illumina HiSeq 2000 short reads. High population diversity, repeat regions, and sequencing errors are among the major factors contributing to the computational challenge associated with metagenomic assembly (reviewed by [46]). These factors reduce the quality of the assembly as reflected, for example, in shorter contig lengths, chimeric contigs and contigs containing strand inversions. These in turn, directly and negatively impact MetaEuk. Shorter contigs limit its ability to discover multi-exon protein-coding genes as it searches for them within a contig. In addition, predictions on contig edges can be partial, which is more likely to happen in a highly fragmented assembly. By dividing each of the benchmark scaffolds to contigs whose lengths were drawn at random based on the length distribution of the Tara Ocean contigs, we showed that while MetaEuk retains its overall sensitivity to detect protein-coding genes even under conditions of increasing evolutionary distance between the query organism and the target reference database, the completeness of its predictions is reduced. We thus expect MetaEuk to benefit from future improvements in assembly algorithms, higher sequencing coverage, and long-read sequencing technology [47–50].

In addition to developing MetaEuk, we generated two useful resources for the analysis of eukaryotes as part of this study. The first is the comprehensive protein profile database, which was computed using protein sequences

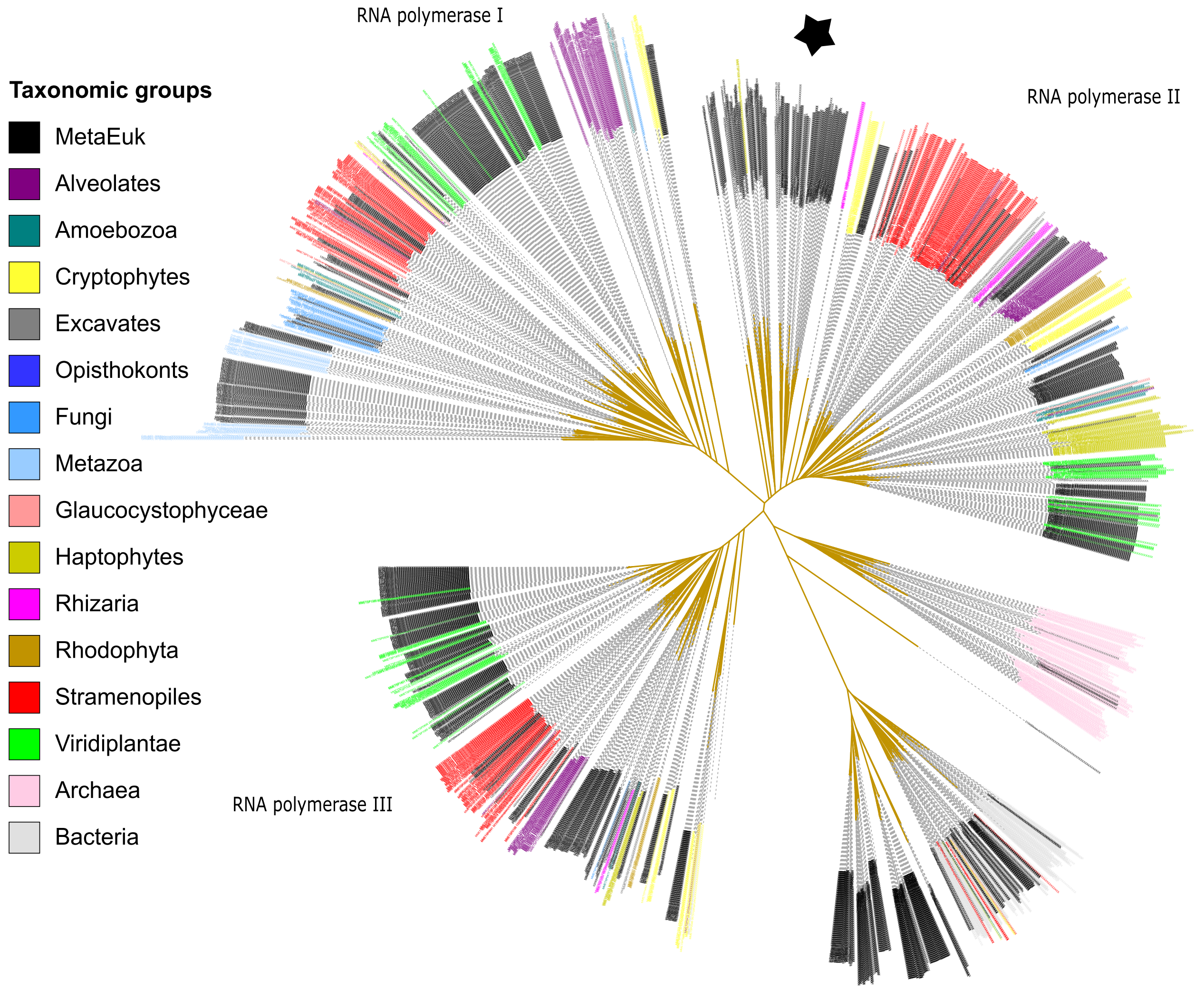


Fig. 7 Diversity of MetaEuk marine eukaryotic proteins. Homologous sequences of the large subunits of RNA polymerases of 985 species as well as 1076 MetaEuk marine proteins were collected and a maximum-likelihood tree was computed based on their alignment. MetaEuk sequences (black) expand major eukaryotic lineages, including deeply rooted supergroups (denoted with star)

from three sources: MERC, MMETSP, and Uniclust50. With ~88 million records, it is the largest profile database focused on eukaryotes to date. Since MERC was assembled from the Tara Oceans metatranscriptomic data, we expected it to be a valuable resource for discovering protein-coding genes in the same environment. Indeed, we found that the majority of MetaEuk predictions (77%) were based on MERC protein profiles. Furthermore, the high fraction of MERC-based predictions that could not be assigned a taxonomic label (63%) demonstrates the uniqueness of this resource.

The second resource is the MetaEuk marine protein collection, which is available on our search web server (https://search.mmseqs.com) for easy investigation [51]. Using a phylogenetic marker protein, we showed that this collection contains proteins spanning major eukaryotic lineages, including supergroups with very few available genomes. Over 33% of these proteins have less than 60% sequence identity to the representative reference proteins that were used to predict them, indicating their diversity with respect to the reference database. Unlike the MERC and MMETSP proteins, MetaEuk proteins are predicted in the context of genomic contigs. This allows us to learn of the number of putative exons that code for them as well as to examine them together with other proteins on the same contig. The latter is useful for conferring taxonomic annotations to unlabeled predictions on the same contig as well as for detecting complex functional modules, by searching for co-occurrences of the module's proteins on the same contig.

As was demonstrated by the challenge of assigning taxonomy to highly diverged eukaryotic proteins, the paucity of eukaryotic sequences in reference databases is currently a major limitation in the study of eukaryotes.

Thus, we expect the resources produced in this study and further analyses of eukaryotic metagenomic data using MetaEuk to produce a more comprehensive description of the tree of life [16, 52–54].

Conclusions

MetaEuk is a sensitive reference-based algorithm for large-scale discovery of protein-coding genes in eukaryotic metagenomic data. Applying MetaEuk to large metagenomic datasets is expected to significantly enrich our databases with highly diverged eukaryotic protein-coding genes. By adding sequences from under-sampled eukaryotic lineages, we can improve sequence homology searches, protein profile computation and thereby homology-based function annotation, template-based, and even de novo protein structure prediction [55, 56]. These, in turn will allow for further exploration of eukaryotic activity in various environments [57].

Methods

MetaEuk algorithm

Code and resources availability

The MetaEuk source code, compilation instructions, and a brief user guide are available from https://github.com/soedinglab/metaeuk under the GNU General Public License v3.0. The resources produced during this study are available from http://wwwuser.gwdg.de/~compbiol/metaeuk/.

Putative exons compatibility

In the first two stages of the MetaEuk algorithm, all possibly coding protein fragments are translated from the input contigs. We scan each contig in six frames and extract the fragments between stop codons. These fragments are queried against the reference target database using MMseqs2. A set of fragments from the same contig and strand that have local matches to the same specific target *T* define a set of putative exons. We say two putative exons P_i and P_j from the same set are compatible with each other if they can be joined together to a multi-exon protein.

Each P_i is associated with four coordinates: the amino-acid position on *T* from which the match to P_i starts (P_i^{ST}) and ends (P_i^{ET}); the nucleotide position on the contig from which the translation of P_i starts (P_i^{SC}) and ends (P_i^{EC}). We require a match of at least 10 amino acids (a minimal exon length). We consider putative exons P_i and P_j with $P_i^{\mathrm{ST}} < P_j^{\mathrm{ST}}$ as compatible on the plus strand if:

(1) their order on the contig is the same as on the target: $P_i^{\mathrm{SC}} < P_j^{\mathrm{SC}}$;
(2) the distance between them on the contig is at least the length of a minimal intron but not more than the length of a maximal intron: $15 \leq (P_j^{\mathrm{SC}} - P_i^{\mathrm{EC}}) \leq 10,000$;
(3) their matches to *T* should not overlap. In practice, we allow for a short overlap to account for alignment errors: $(P_j^{\mathrm{ST}} - P_i^{\mathrm{ET}}) \geq -10$.

In case P_i and P_j are on the negative strand, we modify conditions (1) and (2) accordingly:

$$(1) P_i^{\mathrm{SC}} > P_j^{\mathrm{SC}};$$

$$(2) 15 \leq \left(P_i^{\mathrm{EC}} - \mathrm{P}_j^{\mathrm{SC}}\right) \leq 10,000.$$

Since the adjustment of conditions to the minus strand is straightforward, in the interest of brevity, we focus solely on the plus strand in the following text.

We say a set of $k > 1$ putative exons is compatible if, when ordered by their P_i^{ST} values, each pair of consecutive putative exons is compatible. (A set of a single exon is always compatible.)

Bit-score and *E*-value computation

A set of *k* compatible putative exons defines a pairwise protein alignment to the target *T*. This alignment is the concatenation of the ordered local alignments of all putative exons to *T*. Between each consecutive putative pair of exons P_i and P_{i+1} there might be unmatched amino acids in *T* or there might be a short overlap of their matches to *T*. We denote the number of unmatched amino acids between P_i and P_{i+1} as l_i, which can take a negative value in case of an overlap. MetaEuk computes the bit-score of the concatenated pairwise alignment $S(P_{\mathrm{set}}, T)$ by summing the individual Karlin-Altschul [58] bit-scores $S(P_i, T)$ of the putative exons to *T* and penalizing for unmatched or overlapping amino acids in *T* as follows:

$$^{(3)} S(P_{\mathrm{set}}, T) = \sum_{i=1}^{k} S(P_i, T) + \sum_{i=1}^{k-1} C(l_i) + \log_2(k!)$$

where the penalty function is $C(l_i) = -|l_i|$ for $l_i \neq 1$ and 0 if $l_i = 1$. The last term rewards the correct ordering of the *k* exons.

An *E*-value is the expected number of matches above a given bit-score threshold. Since for each contig, at most one gene call is reported per strand and target in the reference database, the *E*-value takes into account the number of amino acids in the reference database *D* and the search on two strands:

$$(4) E\text{-}\mathrm{Value}(P_{\mathrm{set}}, T) = 2 \times D \times 2^{-S(P_{\mathrm{set}}, T)}$$

Dynamic programming

Given a set of *n* putative exons and their target, MetaEuk finds the set of compatible exons with the highest combined bit-score. First, all putative exons are sorted by their start on the contig, such that $P_1^{\mathrm{SC}} \leq \ldots \leq P_n^{\mathrm{SC}}$. The dynamic programming computation iteratively computes vectors S, k, and b from their first entry 1 to their n^{th}. The entry S_i holds the score of the best exon alignment ending in exon i and k_i holds the number of exons in that set. Once the maximum score is found, the exon alignment is back traced using b, in which entry b_i holds the index of the aligned exon preceding exon i (0 if i is the first aligned exon). Using the following values:

$$(5)\, S_0 = 0; k_0 = 0; b_0 = 0$$

all putative exons P_j are examined according to their order and the score vector is updated:

$$(6)\, S_j = \max_i \left(S_i + S\left(P_j, T\right) + C\left(l_j^i\right) + \log_2(k_i + 1) | 0 \leq i < j, i \text{ compatible with } j\right)$$

k_j and b_j are updated accordingly. The terms $\log_2(k_i + 1)$ add up to the score contribution $\sum_{i=1}^{k} \log_2(i) = \log_2(k!)$ and the transition 0 to j is defined as compatible with $C(l_j^0) = 0$ for all j. The optimal exon set is then recovered by tracing back from the exon with the maximal score. This dynamic programming procedure has time complexity of $O(n^2)$.

Clustering gene calls to reduce redundancy

MetaEuk assigns a unique identifier to each extracted putative protein fragment (stage 1 in Fig. 1). A MetaEuk exon refers to the part of a fragment that matched some target T (stage 2 in Fig. 1, tinted background) and has the same identifier as the fragment. Two calls that have the same exon identifier in their exon set are said to share an exon. MetaEuk reduces redundancy by clustering calls that share an exon (stage 4 in Fig. 1) and selecting a representative call as the gene prediction of each cluster. To that end, all N MetaEuk calls from the same contig and strand combination are ordered according to the contig start position of their first exon. Since this order can include equalities, they are sub-ordered by decreasing number of exons. The first cluster is defined by the first call, which serves as its tentative representative. Let m be the last contig position of the last exon of this representative. Each of the following calls is examined so long as its start position is smaller than m (i.e., it overlaps the representative on the contig). If that call shares an exon with the representative, it is assigned to its cluster. In the next iteration, the first unassigned call serves as representative for a new cluster and the following calls are examined in a similar manner, adding unassigned calls to the cluster in case they share an exon with the representative. The clustering ends with the assignment of all calls. At this stage, the final prediction is the call with the highest score in each cluster. This greedy approach has time complexity of $O(N \times \log(N) + N \times A)$, where A is the average number of calls that overlap each representative on the contig. Since in practice, $A \ll N$, the expected time complexity is $O(N \times \log(N))$.

Resolving same-strand overlapping predictions

After the redundancy reduction step, MetaEuk sorts all predictions on the same contig and strand according to their *E-value* . It examines the sorted list and retains predictions only if they do not overlap any preceding predictions on the list.

Benchmark datasets

The UniRef90 database was obtained on March 2018. The annotated information of *Schizosaccharomyces pombe* (GCA_000002945.2), *Acanthamoeba castellanii str. Neff* (GCA_000313135.1), *Babesia bigemina* (GCA_000981445.1), *Phytomonas sp. isolate* EM1 (GCA_000582765.1), Nucleomorph of *Lotharella oceanica* (GCA_000698435.2), *Phaeodactylum tricornutum* (GCA_000150955.2), and *Aspergillus nidulans* (GCA_000149205.2) were downloaded from the NCBI genome assembly database (March–September 2018). This information included the genomic scaffolds, annotated protein sequences, and GFF3 files containing information about the locations of annotated proteins and other genomic elements. MetaEuk (Github commit 4714106, MMseqs2 submodule version ebb16f3) was run with the following parameters: -e 100 (a lenient maximal *E*-value of a putative exon against a target protein), --metaeuk-eval 0.0001 (a stricter maximal cutoff for the MetaEuk *E-value* after joining exons into a gene call), --metaeuk-tcov 0.6 (a minimal cutoff for the ratio between the MetaEuk protein and the target), and --min-length 20, requiring putative exon fragments of at least 20 codons and default MMseqs2 search parameters.

Mapping benchmark predictions to annotated proteins

For each annotated protein, we listed the contig start and end coordinates of the coding part (CDS) of each of its exons. The lowest and highest of these coordinates were considered as the boundaries of the annotated protein, and the stretch between them as its “global” contig length. Similarly, we listed these coordinates and computed the boundaries and global contig length for each MetaEuk prediction. A MetaEuk prediction was globally mapped to an annotated protein if the overlap computed based on their boundaries was at least 80% of the global contig length of either of them and if, in addition, the alignment of their protein sequences mainly consisted of identical amino acids or gaps (i.e., less than 10%

mismatches). These criteria allow mapping MetaEuk predictions to an annotated protein, even if they miss some of its exons. Next, we computed the exon level mapping for all globally mapped pairs of MetaEuk predictions and annotated proteins. To that end, we compared their lists of exon contig coordinates. If an exon predicted by MetaEuk covered at least 80% of the contig length of an annotated protein's exon, we considered the annotated exon as "covered" by the MetaEuk prediction.

Generating typical metagenomic contig lengths

In order to evaluate MetaEuk's performance on contigs with a length distribution typical for assemblies from metagenomic samples, we recorded the lengths of the assembled contigs used for the analysis described in the "Tara Oceans dataset" section. The 1,351,204 contigs had a minimal length of 5002 bps, 1st quartile of 5661 bps, median of 6763 bps, 3rd quartile of 9020 bps, and a maximal length of 1,524,677 bps. We divided each annotated scaffold into contigs of lengths that were randomly sampled from these recorded lengths. This resulted in 1392, 5061, 1816, 2095, 80, 3153, and 3273 contigs for *S. pombe*, *A. castellanii*, *Phytomonas sp. isolate* EM1, nucleomorph of *L. oceanica*, *P. tricornutum*, and *A. nidulans*, respectively. MetaEuk was run on these contigs in the same way as on the original scaffolds. Since each of the new contigs corresponded to specific locations on the original scaffolds, we could carry out all benchmark assessments, which relied on mapping between MetaEuk predictions and annotated proteins.

Tara Oceans dataset

The 912 metagenomic SRA experiments associated with accession number PRJEB4352 were downloaded from the SRA (August–September 2018). The reads of each experiment were trimmed to remove adapters and low quality sequences using trimmomatic-0.38 [59] with parameters ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36 (SE for single-end samples). The resulting reads were then assembled with MEGAHIT [38] with default parameters. Contigs of at least 5 kbp in length were classified as eukaryotic/non-eukaryotic using EukRep [27], which is trained to be highly sensitive to detecting eukaryotic contigs. MetaEuk was run on the contigs classified as eukaryotic with parameters: -e 100, --metaeuk-eval 0.0001, --min-ungapped-score 35, --min-exon-aa 20, --metaeuk-tcov 0.6, --min-length 40, --slice-search (profile mode), and default MMseqs2 search parameters.

Taxonomic assignment to predictions and contigs

We used MMseqs2 to query the MetaEuk marine protein collection against two taxonomically annotated datasets: Uniclust90 and the MMETSP protein dataset. Taxonomic labels associated with each of the MMETSP identifiers were downloaded from the NCBI website (BioProject PRJNA231566). We retained the hit with the highest bit-score value for each prediction if it had an *E*-value smaller than 10^{-5}. In addition, we examined the sequence identity between the MetaEuk prediction and the target in order to determine the rank of the taxonomic assignment. Similarly to [20], we used the following sequence identity cutoffs: > 95% (species), > 80% (genus), > 65% (family), > 50% (order), > 40% (class), > 30% (phylum), > 20% (kingdom). Lower values were assigned at the domain level. The predictions on each contig were examined and the best-scoring one was used to confer taxonomic annotation to that contig. The assignment was visualized using Krona [60].

Phylogenetic tree reconstruction

We constructed the tree using the large subunit of RNA polymerases as a universal marker. This subunit contains five RNA_pol_Rpb domains (Pfam IDs: pf04997, pf00623, pf04983, pf05000, pf04998). As detailed below, protein sequences that contained all five domains in the right order were obtained in January–November 2019 from six sources to construct the multiple sequence alignment and tree. The sources were as follows: (1) 75 sequences of the OrthoMCL [61] group OG5_127924. The four-letter taxonomic codes of these sequences were converted to NCBI scientific names, based on information from the OrthoMCL website (http://orthomcl.org/orthomcl/getDataSummary.do). (2) 36 reviewed eukaryotic sequences were downloaded from UniProt [36]. These were used to distinguish between eukaryotic RNA polymerase I (8 sequences), eukaryotic RNA polymerase II (16 sequences), and eukaryotic RNA polymerase III (12 sequences). We then ran an MMseqs2 profile search against the Pfam database (with parameters: -k 5, -s 7) with several query sets and retained results in which all five domains were matched in the right order with a maximal *E*-value of 0.0001. This allowed us to add the following sources: (3) 674 MMETSP proteins, (4) 100 archaeal proteins, and (5) 100 bacterial proteins. For datasets (4, 5), we first downloaded candidate proteins from the UniProt database by searching for the five domains and restricting taxonomy: archaea (bacteria). We then ran the previously described search procedure and randomly sampled exactly 100 proteins from each group that matched the criterion. (6) 1076 MetaEuk predictions. The joint set of 2061 sequences was aligned using MAFFT v7.407 [44] and a phylogenetic tree was reconstructed by running RAxML v8 [45]. Tree visualization was performed in iTOL [62].

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s40168-020-00808-x.

Additional file 1: Supplementary Figure S1. MetaEuk predictions by number of exons and exons length. MetaEuk was run on a benchmark of seven eukaryotic unicellular organisms. (A) The fraction of multi-exon MetaEuk predictions is similar to the fraction of annotated multi-exon proteins (Table 1). (B) Single-exon predictions tend to have longer putative exons than multi-exon predictions. **Supplementary Figure S2. MetaEuk target coverage.** The protein sequence of each MetaEuk prediction was aligned to the UniRef90 target, which was used to produce the prediction. The level of target coverage was measured while recording the mapping status of the MetaEuk prediction with respect to the annotations of the benchmark organism: mapped to an annotated protein ("prot"), overlap of at least ten nucleotides with an annotated protein on the opposite strand ("prot. on opp. strand"), prediction on a scaffold for which no NCBI annotations were given ("unannot. scaff.") and all other predictions ("NA"). In all cases, most targets were highly covered by their MetaEuk prediction. **Supplementary Figure S3. MetaEuk E-values and bit-scores.** MetaEuk was run on a benchmark of seven eukaryotic unicellular organisms. The (A) E-values and (B) bit-scores computed between each predicted protein and its target by MetaEuk were compared to those computed by the Smith-Waterman algorithm. The Spearman rho values indicate high correlation for all benchmark organisms. **Supplementary Figure S4. MetaEuk evaluation on typical metagenomic contig lengths.** The annotated scaffolds of each of the organism in Table 1 were randomly divided into shorter contigs, following typical lengths of a metagenomics analysis (see Methods). Since each of the new contigs corresponds to locations on the original scaffolds, MetaEuk predictions on these contigs could be mapped to annotated proteins. (A) Conditions of increasing evolutionary divergence were simulated by excluding gene calls based on their sequence identity to their target. Sensitivity is the fraction of annotated proteins from the query genome to which a MetaEuk prediction was mapped. (B) Fraction of exons covered by MetaEuk (color saturation). The number of MetaEuk predictions is indicated on top of each bar. (C) In an annotation-dependent precision estimation MetaEuk predictions that mapped to an annotated protein were considered as "true" and the rest as "false". (D) Fraction of annotated protein-coding genes that were split by MetaEuk into two (dark grey) or three (black) different predictions. (E) Comparison of the E-values computed by MetaEuk and by the Smith-Waterman algorithm for *A. castellani* proteins. **Supplementary Figure S5. Contribution of reference datasets to MetaEuk predictions.** Profiles computed based on clusters of MERC, MMETSP and Uniclust50 proteins served as the reference database for the MetaEuk run on the Tara Oceans contigs. MERC, MMETSP and Uniclust50 contributed 77.4%, 5.7% and 4.3% of the predictions, respectively. The rest of the predictions were based on mixed-dataset clusters.

Acknowledgements

We thank Dr. David Burstein from Tel Aviv University for his helpful insights concerning the phylogenetic analyses and Dr. Christian Woehle from Christian-Albrechts University Kiel for his comments on the manuscript.

Authors' contributions

ELK and JS have designed the MetaEuk algorithm, benchmark, and biological application. ELK and MM have developed the algorithm. ELK has analyzed the benchmark and Tara Oceans data. ELK and MM have generated the figures. ELK, JS, and MM have drafted the manuscript. All authors read and approved the final manuscript.

Funding

ELK is a recipient of a FEBS long-term fellowship and is an EMBO non-stipendiary long-term fellow. This work was supported by the EU's Horizon 2020 Framework Programme (Virus-X, grant 685778).

Availability of data and materials

The datasets generated and/or analyzed during the current study are available in http://wwwuser.gwdg.de/~compbiol/metaeuk/.

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Competing interests

The authors declare that they have no competing interests.

Received: 15 November 2019 Accepted: 14 February 2020
Published online: 03 April 2020

References

1. Lentendu G, Hübschmann T, Müller S, Dunker S, Buscot F, Wilhelm C. Recovery of soil unicellular eukaryotes: an efficiency and activity analysis on the single cell level. J Microbiol Methods. 2013;95:463–9.
2. Keeling PJ, del Campo J. Marine protists are not just big bacteria. Curr Biol. 2017;27:R541–9.
3. Parfrey LW, Walters WA, Knight R. Microbial eukaryotes in the human microbiome: ecology, evolution, and future directions. Front Microbiol. 2011;2:153.
4. Parfrey LW, Walters WA, Lauber CL, Clemente JC, Berg-Lyons D, Teiling C, et al. Communities of microbial eukaryotes in the mammalian gut within the context of environmental eukaryotic diversity. Front Microbiol. 2014;5.
5. Massana R. Eukaryotic picoplankton in surface oceans. Annu Rev Microbiol. 2011;65:91–110.
6. Flórez LV, Biedermann PHW, Engl T, Kaltenpoth M. Defensive symbioses of animals with prokaryotic and eukaryotic microorganisms. Nat Prod Rep. 2015;32:904–36.
7. Douglas AE. Symbiosis as a general principle in eukaryotic evolution. Cold Spring Harb Perspect Biol. 2014;6:a016113.
8. Field CB, Behrenfeld MJ, Randerson JT, Falkowski P. Primary production of the biosphere: integrating terrestrial and oceanic components. Science. 1998;281:237–40.
9. Jardillier L, Zubkov MV, Pearman J, Scanlan DJ. Significant CO2 fixation by small prymnesiophytes in the subtropical and tropical northeast Atlantic Ocean. ISME J. 2010;4:1180–92.
10. Woehle C, Roy A-S, Glock N, Wein T, Weissenbach J, Rosenstiel P, et al. A novel eukaryotic denitrification pathway in Foraminifera. Curr Biol. 2018;28:2536–2543.e5.
11. Michalak I, Chojnacka K. Algae as production systems of bioactive compounds. Eng Life Sci. 2015;15:160–76.
12. Falaise C, François C, Travers M-A, Morga B, Haure J, Tremblay R, et al. Antimicrobial compounds from eukaryotic microalgae against human pathogens and diseases in aquaculture. Mar Drugs. 2016;14:159.
13. Leray M, Knowlton N. DNA barcoding and metabarcoding of standardized samples reveal patterns of marine benthic diversity. Proc Natl Acad Sci U S A. 2015;112:2076–81.
14. Pawlowski J. The new micro-kingdoms of eukaryotes. BMC Biol. 2013;11:40.
15. Lax G, Eglit Y, Eme L, Bertrand EM, Roger AJ, Simpson AGB. Hemimastigophora is a novel supra-kingdom-level lineage of eukaryotes. Nature. 2018;564:410–4.
16. Burki F. The eukaryotic tree of life from a global phylogenomic perspective. Cold Spring Harb Perspect Biol. 2014;6:a016147.
17. Keeling PJ, Burki F, Wilcox HM, Allam B, Allen EE, Amaral-Zettler LA, et al. The Marine Microbial Eukaryote Transcriptome Sequencing Project (MMETSP): illuminating the functional diversity of eukaryotic life in the oceans through transcriptome sequencing. PLoS Biol. 2014;12:e1001889.
18. Bik HM, Porazinska DL, Creer S, Caporaso JG, Knight R, Thomas WK. Sequencing our way towards understanding global eukaryotic biodiversity. Trends Ecol Evol. 2012;27:233–43.
19. Sunagawa S, Coelho LP, Chaffron S, Kultima JR, Labadie K, Salazar G, et al. Structure and function of the global ocean microbiome. Science. 2015;348:1261359.
20. Carradec Q, Pelletier E, Da Silva C, Alberti A, Seeleuthner Y, Blanc-Mathieu R, et al. A global ocean atlas of eukaryotic genes. Nat Commun. 2018;9:373.

21. Escobar-Zepeda A, Vera-Ponce de León A, Sanchez-Flores A. The road to metagenomics: from microbiology to DNA sequencing technologies and bioinformatics. Front Genet. 2015;6:348.
22. Majaneva M, Hyytiäinen K, Varvio SL, Nagai S, Blomster J. Bioinformatic amplicon read processing strategies strongly affect eukaryotic diversity and the taxonomic composition of communities. PLoS One. 2015;10:e0130035.
23. Stanke M, Morgenstern B. AUGUSTUS: a web server for gene prediction in eukaryotes that allows user-defined constraints. Nucleic Acids Res. 2005;33:W465–7.
24. Hoff KJ, Lange S, Lomsadze A, Borodovsky M, Stanke M. BRAKER1: unsupervised RNA-seq-based genome annotation with GeneMark-ET and Augustus. Bioinformatics. 2016;32:767–9.
25. Hoff KJ, Stanke M. WebAUGUSTUS--a web service for training AUGUSTUS and predicting genes in eukaryotes. Nucleic Acids Res. 2013;41:W123–8.
26. Korf I. Gene finding in novel genomes. BMC Bioinformatics. 2004;5:59.
27. West PT, Probst AJ, Grigoriev IV, Thomas BC, Banfield JF. Genome-reconstruction for eukaryotes from complex natural microbial communities. Genome Res. 2018;28:569–80.
28. Sedlar K, Kupkova K, Provaznik I. Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. Comput Struct Biotechnol J. 2017;15:48–55.
29. Lu YY, Chen T, Fuhrman JA, Sun F. COCACOLA: binning metagenomic contigs using sequence COmposition, read CoverAge, CO-alignment and pairedend read LinkAge. Bioinformatics. 2017;33:791–8.
30. Yu G, Jiang Y, Wang J, Zhang H, Luo H. BMC3C: binning metagenomic contigs using codon usage, sequence composition and read coverage. Bioinformatics. 2018;34:4172–9.
31. Gelfand MS, Mironov AA, Pevzner PA. Gene recognition via spliced sequence alignment. Proc Natl Acad Sci U S A. 1996;93:9061–6.
32. Gotoh O. Direct mapping and alignment of protein sequences onto genomic sequence. Bioinformatics. 2008;24:2438–44.
33. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol. 2017;35:1026-8.
34. Kumar A. An overview of nested genes in eukaryotic genomes. Eukaryot Cell. 2009;8:1321–9.
35. Benson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Ostell J, Pruitt KD, et al. GenBank. Nucleic Acids Res. 2018;46:D41–7.
36. Bateman A, Martin MJ, O'Donovan C, Magrane M, Alpi E, Antunes R, et al. UniProt: the universal protein knowledgebase. Nucleic Acids Res. 2017;45:D158–69.
37. Delmont TO, Quince C, Shaiber A, Esen ÖC, Lee ST, Rappé MS, et al. Nitrogen-fixing populations of Planctomycetes and Proteobacteria are abundant in surface ocean metagenomes. Nat Microbiol. 2018;3:804–13.
38. Li D, Liu C-M, Luo R, Sadakane K, Lam T-W. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics. 2015;31:1674–6.
39. Mirdita M, von den Driesch L, Galiez C, Martin MJ, Söding J, Steinegger M. Uniclust databases of clustered and deeply annotated protein sequences and alignments. Nucleic Acids Res. 2017;45:D170–6.
40. Steinegger M, Mirdita M, Söding J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. Nat Methods. 2019;16:603–6.
41. Johnson LK, Alexander H, Brown CT. Re-assembly, quality evaluation, and annotation of 678 microbial eukaryotic reference transcriptomes. Gigascience. 2019;8:giy158.
42. Steinegger M, Söding J. Clustering huge protein sequence sets in linear time. Nat Commun. 2018;9:2542.
43. Ren R, Sun Y, Zhao Y, Geiser D, Ma H, Zhou X. Phylogenetic resolution of deep eukaryotic and fungal relationships using highly conserved low-copy nuclear genes. Genome Biol Evol. 2016;8:2683–701.
44. Nakamura T, Yamada KD, Tomii K, Katoh K. Parallelization of MAFFT for large-scale multiple sequence alignments. Hancock J, editor. Bioinformatics. 2018;34:2490–2.
45. Stamatakis A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics. 2014;30:1312–3.
46. Ghurye JS, Cepeda-Espinoza V, Pop M. Metagenomic assembly: overview, challenges and applications. Yale J Biol Med. 2016;89:353–62.
47. Warwick-Dugdale J, Solonenko N, Moore K, Chittick L, Gregory AC, Allen MJ, et al. Long-read viral metagenomics enables capture of abundant and microdiverse viral populations and their niche-defining genomic islands. PeerJ. 2019;7:e6800.
48. Frank JA, Pan Y, Tooming-Klunderud A, Eijsink VGH, McHardy AC, Nederbragt AJ, et al. Improved metagenome assemblies and taxonomic binning using long-read circular consensus sequence data. Sci Rep. 2016;6:25373.
49. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017;27:722–36.
50. Driscoll CB, Otten TG, Brown NM, Dreher TW. Towards long-read metagenomics: complete assembly of three novel genomes from bacteria dependent on a diazotrophic cyanobacterium in a freshwater lake co-culture. Stand Genomic Sci. 2017;12:9.
51. Mirdita M, Steinegger M, Söding J. MMseqs2 desktop and local web server app for fast, interactive sequence searches. Bioinformatics. 2019;35:2856–8.
52. Mann DG, Droop SJM. Biodiversity, biogeography and conservation of diatoms. Hydrobiologia. 1996;336:19–32.
53. Norton TA, Melkonian M, Andersen RA. Algal biodiversity. Phycologia. 1996;35:308–26.
54. de Vargas C, Audic S, Henry N, Decelle J, Mahe F, Logares R, et al. Eukaryotic plankton diversity in the sunlit ocean. Science. 2015;348:1261605.
55. Ovchinnikov S, Park H, Varghese N, Huang P-S, Pavlopoulos GA, Kim DE, et al. Protein structure determination using metagenome sequence data. Science. 2017;355:294–8.
56. Söding J. Big-data approaches to protein structure prediction. Science. 2017;355:248–9.
57. Worden AZ, Allen AE. The voyage of the microbial eukaryote. Curr Opin Microbiol. 2010;13:652–60.
58. Karlin S, Altschul SF. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc Natl Acad Sci U S A. 1990;87:2264–8.
59. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics. 2014;30:2114–20.
60. Ondov BD, Bergman NH, Phillippy AM. Interactive metagenomic visualization in a web browser. BMC Bioinformatics. 2011;12:385.
61. Chen F, Mackey AJ, Stoeckert CJ, Roos DS. OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. Nucleic Acids Res. 2006;34:D363–8.
62. Letunic I, Bork P. Interactive Tree Of Life (iTOL) v4: recent updates and new developments. Nucleic Acids Res. 2019;47:W256–9.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Supplementary Information

MetaEuk – sensitive, high-throughput gene discovery and annotation for large-scale eukaryotic metagenomics

Eli Levy Karin[1*], Milot Mirdita[1], and Johannes Söding[1*]

[1] Quantitative and Computational Biology, Max-Planck Institute for Biophysical Chemistry, 37077 Göttingen, Germany.

* To whom correspondence should be addressed:

Eli Levy Karin, Tel: +49 551 201-2881

E-mail: eli.levy.karin@gmail.com

Johannes Söding, Tel: +49 551 201-2890

E-mail: soeding@mpibpc.mpg.de

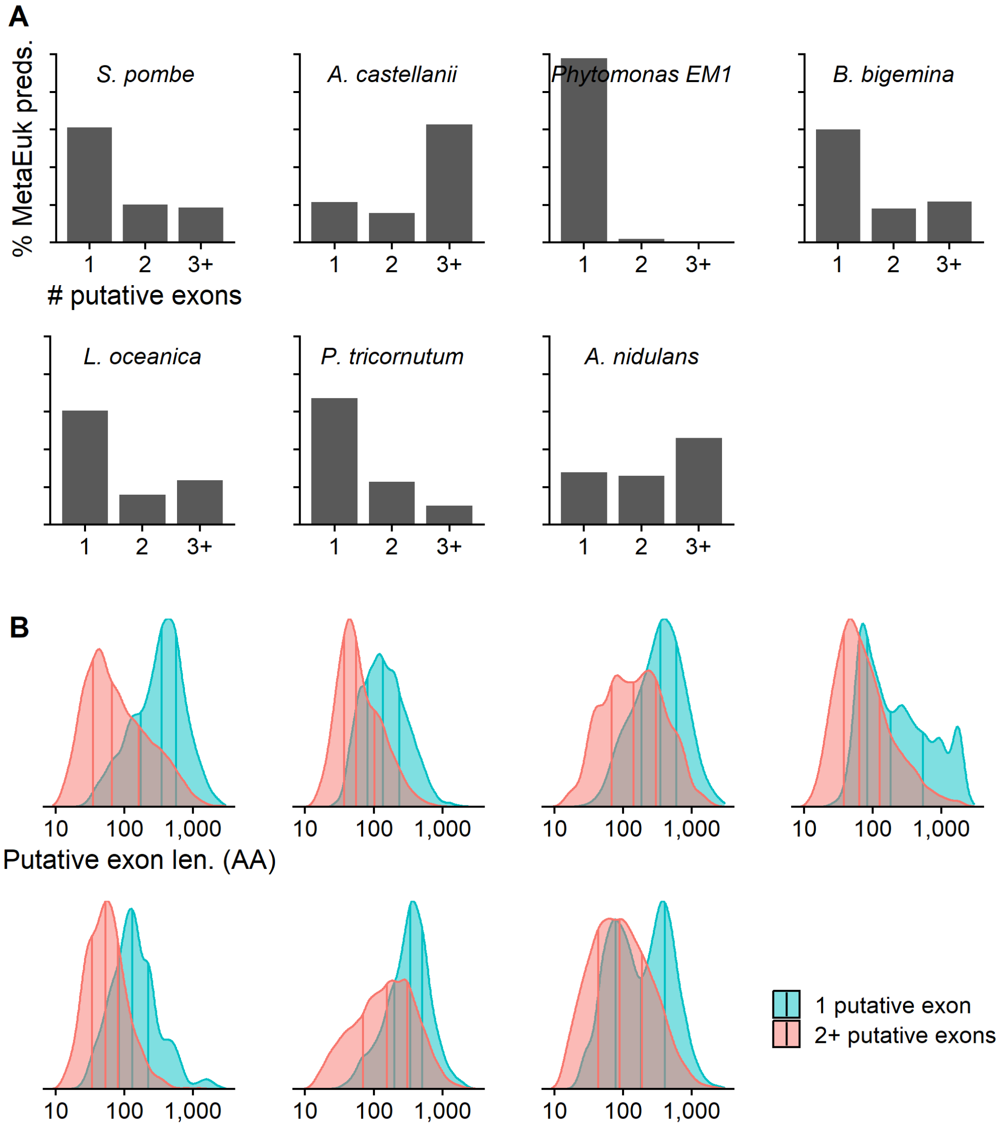
Running title: Metagenomics eukaryotic gene discovery

Keywords: MetaEuk, eukaryotes, homology detection, prediction, annotation, contigs, marine;

Supplementary Figures

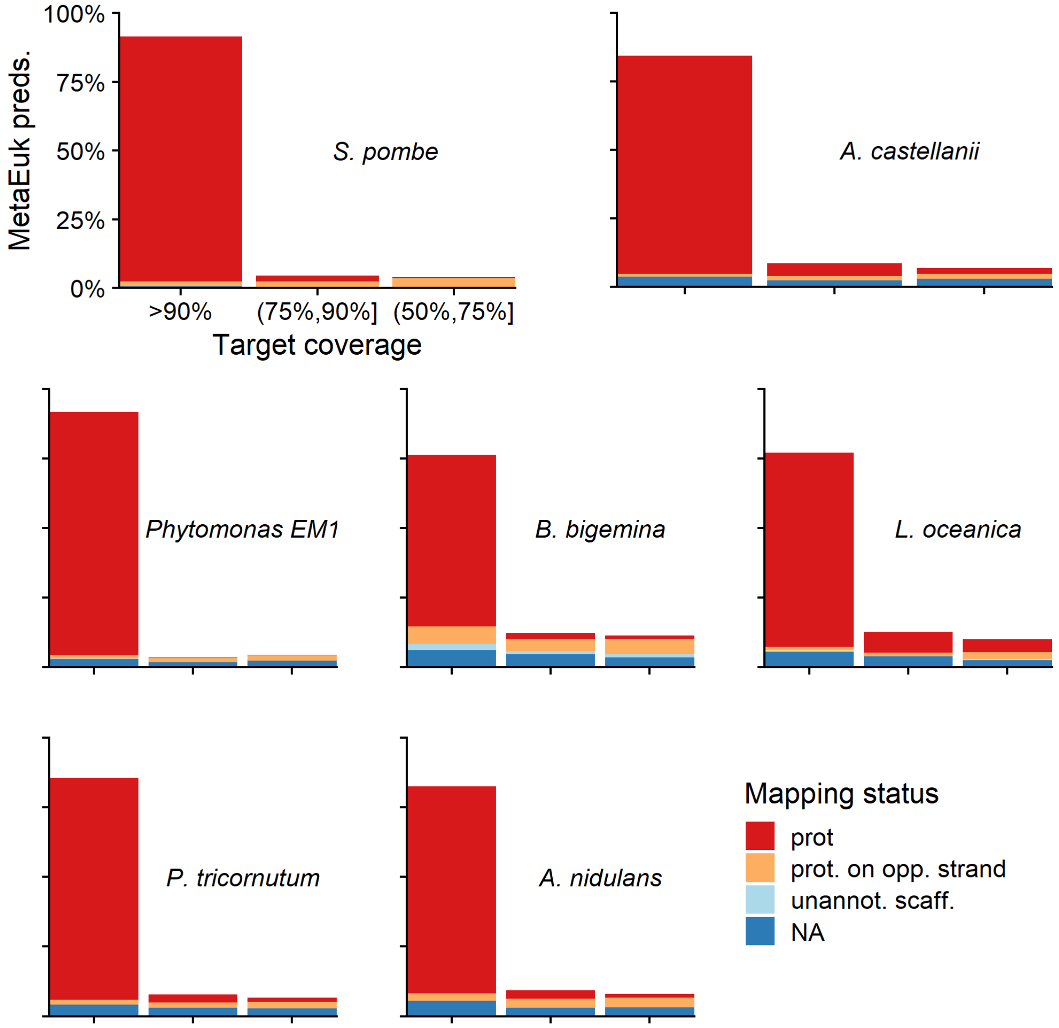
Supplementary Figure 1 – MetaEuk predictions by number of exons and exons length

MetaEuk was run on a benchmark of seven eukaryotic unicellular organisms. (A) The fraction of multi-exon MetaEuk predictions is similar to the fraction of annotated multi-exon proteins (Table 1). (B) Single-exon predictions tend to have longer putative exons than multi-exon predictions.



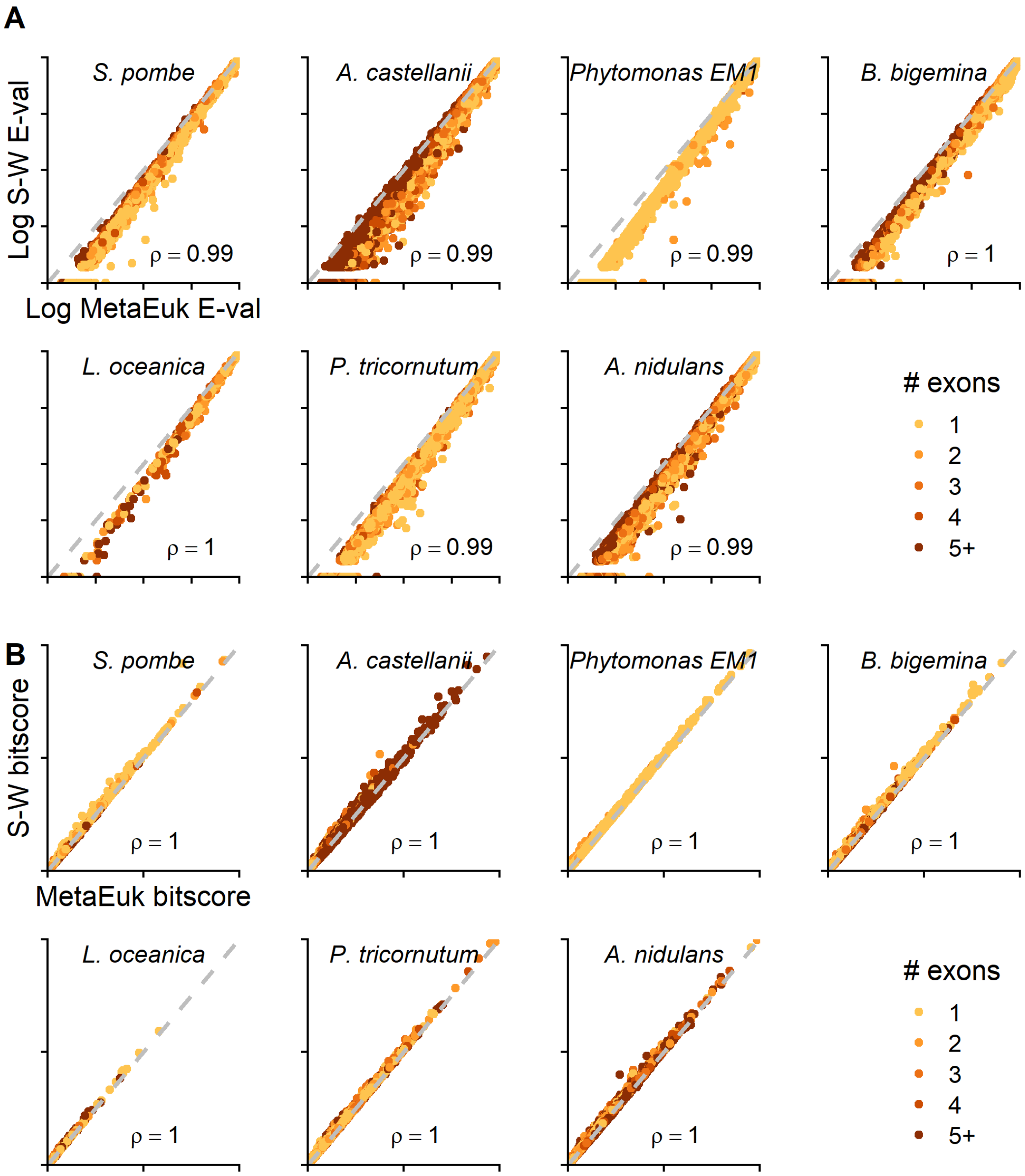
Supplementary Figure 2 – MetaEuk target coverage

The protein sequence of each MetaEuk prediction was aligned to the UniRef90 target, which was used to produce the prediction. The level of target coverage was measured while recording the mapping status of the MetaEuk prediction with respect to the annotations of the benchmark organism: mapped to an annotated protein (“prot”), overlap of at least ten nucleotides with an annotated protein on the opposite strand (“prot. on opp. strand”), prediction on a scaffold for which no NCBI annotations were given (“unannot. scaff.”) and all other predictions (“NA”). In all cases, most targets were highly covered by their MetaEuk prediction.



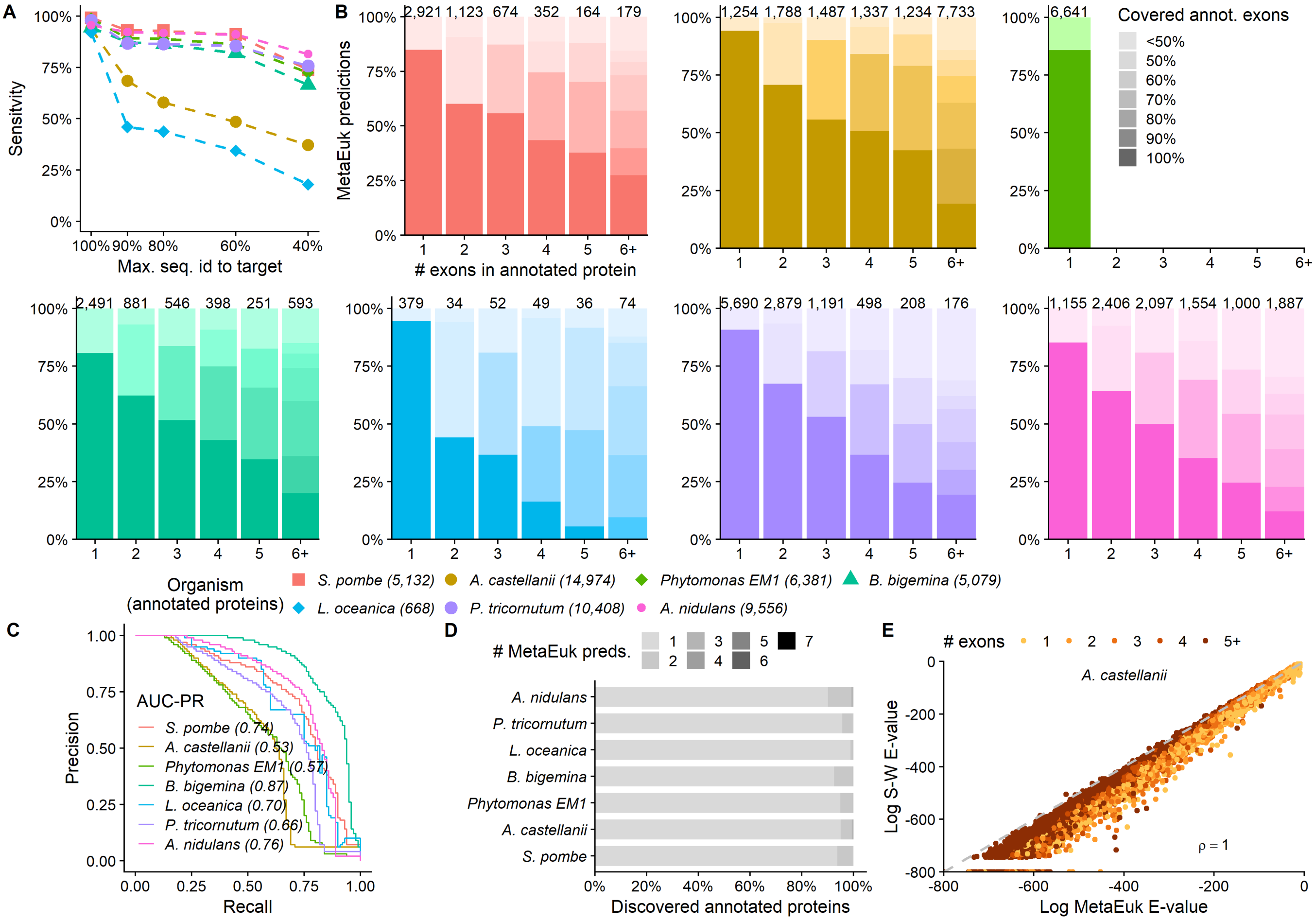
Supplementary Figure 3 – MetaEuk E-values and bit-scores

MetaEuk was run on a benchmark of seven eukaryotic unicellular organisms. The (A) E-values and (B) bit-scores computed between each predicted protein and its target by MetaEuk were compared to those computed by the Smith-Waterman algorithm. The Spearman rho values indicate high correlation for all benchmark organisms.



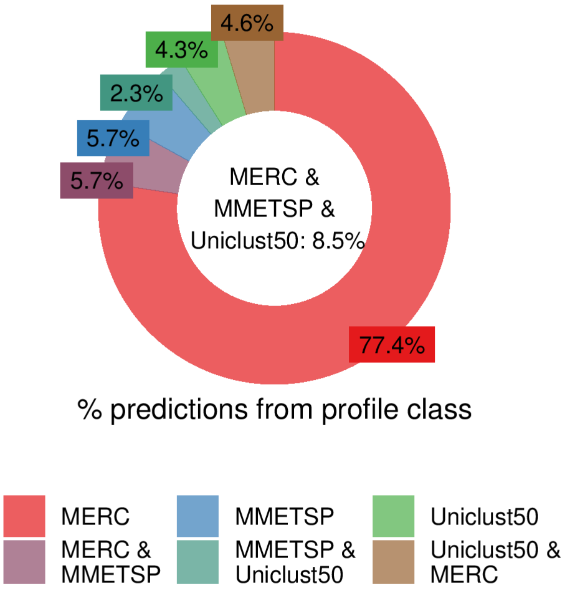
Supplementary Figure 4 – MetaEuk evaluation on typical metagenomic contig lengths

The annotated scaffolds of each of the organism in Table 1 were randomly divided into shorter contigs, following typical lengths of a metagenomics analysis (see Methods). Since each of the new contigs corresponds to locations on the original scaffolds, MetaEuk predictions on these contigs could be mapped to annotated proteins. (A) Conditions of increasing evolutionary divergence were simulated by excluding gene calls based on their sequence identity to their target. Sensitivity is the fraction of annotated proteins from the query genome to which a MetaEuk prediction was mapped. (B) Fraction of exons covered by MetaEuk (color saturation). The number of MetaEuk predictions is indicated on top of each bar. (C) In an annotation-dependent precision estimation MetaEuk predictions that mapped to an annotated protein were considered as “true” and the rest as “false”. (D) Fraction of annotated protein-coding genes that were split by MetaEuk into two (dark grey) or three (black) different predictions. (E) Comparison of the E-values computed by MetaEuk and by the Smith-Waterman algorithm for *A. castellani* proteins.


A
Sensitivity
100%
75%
50%
25%
0%
100% 90% 80% 60% 40%
Max. seq. id to target
B
MetaEuk predictions
# exons in annotated protein
2,921 1,123 674 352 164 179
1,254 1,788 1,487 1,337 1,234 7,733
6,641
Covered annot. exons
<50%
50%
60%
70%
80%
90%
100%
2,491 881 546 398 251 593
379 34 52 49 36 74
5,690 2,879 1,191 498 208 176
1,155 2,406 2,097 1,554 1,000 1,887
1 2 3 4 5 6+
Organism
(annotated proteins)
S. pombe (5,132)
A. castellanii (14,974)
Phytomonas EM1 (6,381)
B. bigemina (5,079)
L. oceanica (668)
P. tricornutum (10,408)
A. nidulans (9,556)
C
Precision
Recall
0.00 0.25 0.50 0.75 1.00
AUC-PR
S. pombe (0.74)
A. castellanii (0.53)
Phytomonas EM1 (0.57)
B. bigemina (0.87)
L. oceanica (0.70)
P. tricornutum (0.66)
A. nidulans (0.76)
D
# MetaEuk preds.
1 2 3 4 5 6 7
A. nidulans
P. tricornutum
L. oceanica
B. bigemina
Phytomonas EM1
A. castellanii
S. pombe
0% 20% 40% 60% 80% 100%
Discovered annotated proteins
E
# exons 1 2 3 4 5+
A. castellanii
ρ = 1
Log S-W E-value
Log MetaEuk E-value
0 -200 -400 -600 -800

Supplementary Figure 5 – Contribution of reference datasets to MetaEuk predictions

Profiles computed based on clusters of MERC, MMETSP and Uniclust50 proteins served as the reference database for the MetaEuk run on the Tara Oceans contigs. MERC, MMETSP and Uniclust50 contributed 77.4%, 5.7% and 4.3% of the predictions, respectively. The rest of the predictions were based on mixed-dataset clusters.



5.4 SpacePHARER: sensitive identification of phages from CRISPR spacers in prokaryotic hosts

Publication:

SpacePHARER: sensitive identification of phages from CRISPR spacers in prokaryotic hosts

R. Zhang, **M. Mirdita**, E. Levy Karin, C. Norroy, C. Galiez, J. Söding†

(†) corresponding author

Bioinformatics (2021), 37, 3364–3366.
Cited 7 times since 4/2021.

Code and software availability

SpacePHARER is available as free open source software (GPLv3) at `spacepharer.soedinglab.org`.

Author contributions

R.Z., E.L.K., C.G. & J.S. designed the SpacePHARER algorithm. R.Z., **M.M.**, J.S. designed the taxonomic assignment in SpacePHARER. R.Z., **M.M.** & C.N. implemented code. R.Z. performed benchmarks. R.Z. & **M.M.** generated figures. R.Z., **M.M.**, E.L.K & J.S. wrote the manuscript.

Bioinformatics, 2021, 1–3
doi: 10.1093/bioinformatics/btab222
Advance Access Publication Date: 1 April 2021
Applications note

Sequence analysis

SpacePHARER: sensitive identification of phages from CRISPR spacers in prokaryotic hosts

Ruoshi Zhang [1], Milot Mirdita [1], Eli Levy Karin [1], Clovis Norroy[1], Clovis Galiez [1,2] and Johannes Söding [1,3,]*

[1]Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany, [2]University of Grenoble Alpes, CNRS, Grenoble INP/Institute of Engineering, LJK, Grenoble, France, and [3]Campus-Institut Data Science (CIDAS), Göttingen, Germany

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on May 15, 2020; revised on March 13, 2021; editorial decision on March 26, 2021; accepted on March 31, 2021

Abstract

Summary: SpacePHARER (CRISPR Spacer Phage–Host Pair Finder) is a sensitive and fast tool for *de novo* prediction of phage–host relationships via identifying phage genomes that match CRISPR spacers in genomic or metagenomic data. SpacePHARER gains sensitivity by comparing spacers and phages at the protein level, optimizing its scores for matching very short sequences, and combining evidence from multiple matches, while controlling for false positives. We demonstrate SpacePHARER by searching a comprehensive spacer list against all complete phage genomes.

Availability and implementation: SpacePHARER is available as an open-source (GPLv3), user-friendly command-line software for Linux and macOS: https://github.com/soedinglab/spacepharer.

Contact: soeding@mpibpc.mpg.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Viruses of bacteria and archaea (phages) are the most abundant biological entities in nature. However, little is known about their roles in the microbial ecosystem and how they interact with their hosts, as cultivating most phages and hosts in the lab is challenging. Many prokaryotes (40% of bacteria and 81% of archaea) possess an adaptive immune system against phages, the Clustered Regularly Interspaced Short Palindromic Repeat (CRISPR)-CRISPR associated (Cas) system (Burstein *et al.*, 2016). After surviving a phage infection, they can incorporate a short DNA fragment (28–42 nt) as a spacer in a CRISPR array. The transcribed spacer will be used with other Cas components for a targeted destruction of future invaders. Some CRISPR-Cas systems require a 2–6 nucleotide long, highly conserved protospacer-adjacent motif (PAM) flanking the viral target to prevent autoimmunity. Multiple spacers targeting the same invader are not uncommon, due to either multiple infection events or the primed spacer acquisition mechanism identified in some CRISPR subtypes. CRISPR spacers have been previously exploited to identify phage–host relationships (Biswas *et al.*, 2013; Dion *et al.*, 2021; Paez-Espino *et al.*, 2016; Shmakov *et al.*, 2017; Stern *et al.*, 2012). These methods compare individual CRISPR spacers with phage genomes using BLASTN (Altschul *et al.*, 1990) and apply stringent filtering criteria, e.g. allowing only up to two mismatches. They are thus limited to identifying very close matches. However, a higher sensitivity is crucial because phage reference databases are very incomplete and often will not contain phages highly similar to those to be identified. To increase sensitivity, (i) we compare protein coding sequences because phage genomes are mostly coding, and, to evade the CRISPR immune response, are under pressure to mutate their genome with minimal changes to the amino acids; (ii) we choose an optimized substitution matrix and gap penalties for short, highly similar proteins; and (iii) we combine evidence from multiple spacers matching to the same phage genome.

2 Materials and methods

Input. SpacePHARER accepts spacer sequences as multiple FASTA files each containing spacers from a single prokaryotic genome or as multiple output files from the CRISPR detection tools PILER-CR (Edgar, 2007), CRT (Bland *et al.*, 2007), MinCED (Skennerton, 2016) or CRISPRDetect (Biswas *et al.*, 2016). Phage genomes are supplied as separate FASTA files or can be downloaded by SpacePHARER from NCBI GenBank (Benson *et al.*, 2013). Optionally, additional taxonomic labels can be provided for spacers or phages to be included in the final report.

Algorithm. SpacePHARER is divided into five steps (Fig. 1A, Supplementary Materials). (0) Preprocess input: scan the phage

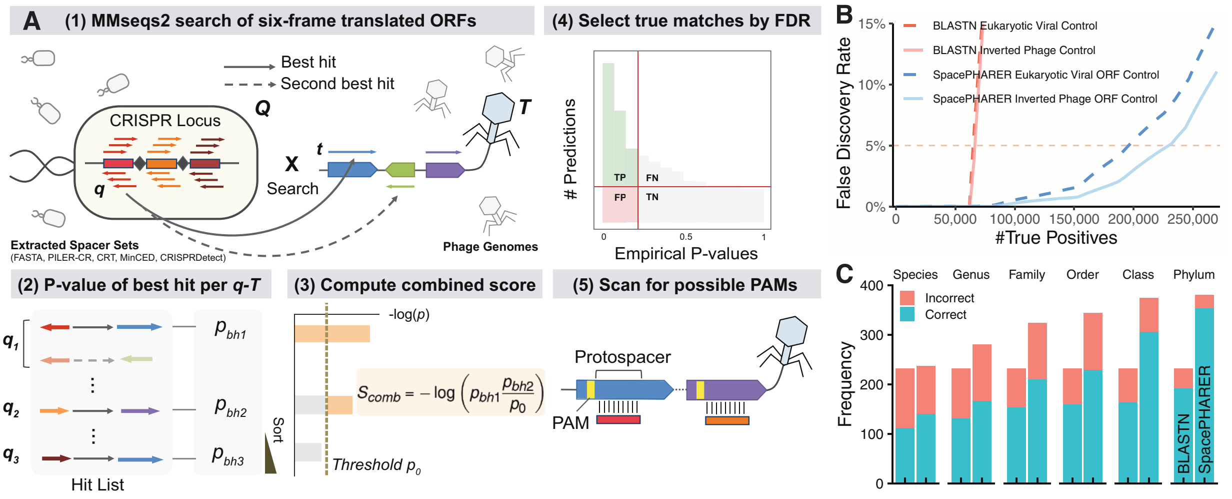

A
(1) MMseqs2 search of six-frame translated ORFs
Best hit
Second best hit
Q
T
CRISPR Locus
q
t
X
Search
Extracted Spacer Sets
(FASTA, PILER-CR, CRT, MinCED, CRISPRDetect)
Phage Genomes
(4) Select true matches by FDR
# Predictions
TP
FN
FP
TN
0
0.5
1
Empirical P-values
B
False Discovery Rate
15%
10%
5%
0%
BLASTN Eukaryotic Viral Control
BLASTN Inverted Phage Control
SpacePHARER Eukaryotic Viral ORF Control
SpacePHARER Inverted Phage ORF Control
0
50,000
100,000
150,000
200,000
250,000
#True Positives
(2) P-value of best hit per q-T
q1
q2
q3
pbh1
pbh2
pbh3
Sort
Hit List
(3) Compute combined score
-log(p)
$S_{comb} = -\log\left(p_{bh1}\frac{p_{bh2}}{p_0}\right)$
Threshold p0
(5) Scan for possible PAMs
Protospacer
PAM
C
Species
Genus
Family
Order
Class
Phylum
Incorrect
Correct
Frequency
400
300
200
100
0
BLASTN
SpacePHARER

Fig. 1. (**A**) SpacePHARER algorithm. A query set Q consists of 6-frame translated ORFs (q) from CRISPR spacers, and a target set T consists of 6-frame translated ORFs (t) of phage proteins. (1) Search all qs against all ts using MMseqs2. Align the q—t hits on nucleotide level and prioritize near-perfect nucleotide hits. (2) For each q—T pair, compute the P-value for the best hit from first-order statistics. (3) Compute score S_{comb} by combining the best-hit P-values from multiple hits between Q and T using a modified truncated-product method. (4) Estimate the FDR by searching a null database. (5) Scan for possible protospacer adjacent motif (PAM). (**B**) Performance comparison between SpacePHARER (blue) and BLASTN (red) using inverted phage sequences (solid lines) or eukaryotic viral ORFs as null set (dashed lines) demonstrated by expected number of true positive (TP) predictions at different false discovery rates (FDRs). (**C**) Performance comparison between BLASTN (left), SpacePHARER using the weighted lowest common ancestor procedure (LCA, right) at FDR = 0.02, evaluated by the number of correct (blue) and incorrect (red) predictions, for all the host predictions made at each taxonomic rank or below. BLASTN hits with >95% sequence identity and query coverage (up to 2 mismatches) were retained.

genome and CRISPR spacers in six reading frames, extract and translate all putative coding fragments of at least 27 nt, with user-definable translation tables. Each query set Q consists of the translated ORFs q of CRISPR spacers extracted from one prokaryotic genome, and each target set T comprises the putative protein sequences t from a single phage. We refer to similar q and t as *hit*, and an identified host-phage relationship Q—T as *match*. (1) Search all q's against all t's using the fast, sensitive MMseqs2 protein search (Steinegger and Söding, 2017), with VTML40 substitution matrix (Müller *et al.*, 2002), gap open cost of 16 and extension cost of 2 (Supplementary Fig. S1). We optimized a short, spaced k-mer pattern for the prefilter stage (10111011) with six informative ('1') positions. In addition, align all q—t hits reported in the previous search on nucleotide level and prioritize near-perfect nucleotide hits (Supplementary Materials). (2) For each q—T pair, compute the P-value for the best hit p_{bh} from first-order statistics. (3) Compute a combined score S_{comb} from best-hit P-values of multiple hits between Q and T using a modified truncated-product method (Supplementary Materials). (4) Compute the false discovery rate (FDR = FP/(TP + FP)) and only retain matches with FDR < 0.05. For that purpose, SpacePHARER is run on a null model database and the fraction of null matches with S_{comb} below a cutoff (empirical P-value) is used to estimate the FDR. (5) Scan 10 nt upstream and downstream of the phage's protospacer for a possible PAM.

Output is a tab-separated text file. Each host-phage match spans two or more lines. The first starts with '#': prokaryote accession, phage accession, S_{comb}, number of hits in the match. Each following line describes an individual hit: spacer accession, phage accession, p_{bh}, spacer start and end, phage start and end, possible 5' PAM—3' PAM, possible 5' PAM—3' PAM on the reverse strand. If requested, the spacer–phage sequence alignments are included. If taxonomic labels are provided, taxonomic reports based on the weighted lowest common ancestor (LCA) procedure described in Mirdita *et al.* (2021) are created for host LCAs of each phage genome or phage LCAs of each spacer as additional tab-separated text files.

3 Results

Datasets. We split a previously published spacer dataset (Shmakov *et al.*, 2017) of 363 460 unique spacers from 30 389 prokaryotic genomes randomly into an optimization set (20%, 6067 genomes) and a test set (80%, 24 322 genomes). The performance of SpacePHARER was evaluated on the spacer test set against a target database of 7824 phage genomes. We used two null databases: 11 304 eukaryotic viral genomes and the inverted translated sequences of the target database. Viral genomes were downloaded from GenBank in 09/2018. The performance of SpacePHARER in Figure 1C was evaluated on a validation dataset of spacers from 1066 bacterial genomes against 809 phage genomes with annotated host taxonomy (Edwards *et al.*, 2016). For each phage, we predicted the host based on the host LCA.

Prediction quality. At FDR = 0.05, SpacePHARER predicted 3 to 4 times more prokaryote-phage matches than BLASTN (Fig. 1B, Supplementary Fig. S2). SpacePHARER predicted the correct host for more phages than BLASTN at all taxonomic ranks, while including most of the BLASTN predictions, at better precision (Fig. 1C, Supplementary Figs S3 and S4). If the host or a close relative of a phage is absent in the database (either because the host is unidentified or the host lacks a CRISPR-Cas system), the predicted host may be correct only at a higher rank than species.

Run time. SpacePHARER took 12 min to process the test dataset on 2 × 6-core Intel E5-2620v3 CPUs, 47 times faster than BLASTN (575 min).

4 Conclusion

SpacePHARER is 1.4 to 4× more sensitive than BLASTN in detecting phage–host pairs, due to searching with protein sequences, optimizing short sequence comparisons, and combining statistical evidence, and it is fast enough to analyze large-scale genomic and metagenomic datasets.

Funding

E.L.K. is a FEBS long-term fellowship recipient and an EMBO non-stipendiary long-term fellow. The work was supported by the ERC's Horizon 2020 Framework Programme ['Virus-X', project no. 685778] and the BMBF CompLifeSci project horizontal4meta.

Conflict of Interest: none declared.

Data availability

The data used to benchmark SpacePHARER and BLASTN are publicly available from ftp://ftp.ncbi.nih.gov/pub/wolf/_suppl/spacerome/ and http://edwards.sdsu.edu/PhageHosts/. The viral genomes were downloaded from NCBI Genbank in 09/2018.

References

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Benson,D.A. *et al.* (2013) GenBank. *Nucleic Acids Res.*, **41**, D36–D42.

Biswas,A. *et al.* (2013) CRISPRTarget: bioinformatic prediction and analysis of crRNA targets. *RNA Biol.*, **10**, 817–827.

Biswas,A. *et al.* (2016) CRISPRdetect: a flexible algorithm to define CRISPR arrays. *BMC Genomics*, **17**, 356.

Bland,C. *et al.* (2007) CRISPR recognition tool (CRT): a tool for automatic detection of clustered regularly interspaced palindromic repeats. *BMC Bioinformatics*, **8**, 209.

Burstein,D. *et al.* (2016) Major bacterial lineages are essentially devoid of CRISPR-Cas viral defence systems. *Nat. Commun.*, **7**, 10613.

Dion,M.B. *et al.* (2021) Streamlining CRISPR spacer-based bacterial host predictions to decipher the viral dark matter. *Nucleic Acids Res.*, **49**(6), 3127–3128.

Edgar,R.C. (2007) PILER-CR: fast and accurate identification of CRISPR repeats. *BMC Bioinformatics*, **8**, 18.

Edwards,R.A. *et al.* (2016) Computational approaches to predict bacteriophage–host relationships. *FEMS Microbiol. Rev.*, **40**, 258–272.

Mirdita,M. *et al.* (2021) Fast and sensitive taxonomic assignment to metagenomic contigs. *Bioinformatics*, btab184.

Müller,T. *et al.* (2002) Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol. Biol. Evol.*, **19**, 8–13.

Paez-Espino,D. *et al.* (2016) Uncovering Earth's virome. *Nature*, **536**, 425–430.

Shmakov,S.A. *et al.* (2017) The CRISPR spacer space is dominated by sequences from species-specific mobilomes. *mBio*, **8**, e01397–17.

Skennerton,C. (2016) Minced – mining CRISPRs in environmental datasets. (15 May 2020, date last accessed). https://github.com/ctSkennerton/minced.

Steinegger,M. and Söding,J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.

Stern,A. *et al.* (2012) CRISPR targeting reveals a reservoir of common phages associated with the human gut microbiome. *Genome Res.*, **22**, 1985–1994.

Supplementary Material for SpacePHARER: Sensitive identification of phages from CRISPR spacers in prokaryotic hosts

Zhang R.,[1] Mirdita M.,[1] Levy Karin E.,[1] Norroy C.,[1] Galiez C.,[1,2] and Söding J.[1,3]

[1] *Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany*
[2] *Univ. Grenoble Alpes, CNRS, Grenoble INP/Institute of Engineering Univ. Grenoble Alpes, Grenoble, France*
[3] *Campus-Institut Data Science (CIDAS), Göttingen, Germany.*

I. ALGORITHM DESCRIPTION

The query spacer set Q has N_q translated ORFs q of CRISPR spacers ($Q = \{q_1, ...q_{N_q}\}$) from one prokaryotic genome. Phage proteome target set T has N_t phage protein sequences t ($T = \{t_1, ...t_{N_t}\}$). These protein sequences are extracted in the input preprocessing step (Step 0) of the algorithm from each spacer set and each phage genome by scanning them in six translational frames. We refer to similarity between q and t as hit, and similarity between Q and T as match. The SpacePHARER algorithm relies on a statistic for the combination of hits between a spacer sequence set and a phage protein sequence set. The idea is that combining together several sub-significant hits (due to weak homologies or the typical length of spacers) can be highly informative and result in a significant match. Steps 2 and 3 of the algorithm test if the pairwise P-values of the best hit of sequences in the query set with those in the target set are due to homologous relationships or entirely due to chance.

A. (1) MMseqs2 protein-level search

The SpacePHARER algorithm first searches all q's against all t's using the fast, sensitive MMseqs2 protein-level search [10], with VTML40 substitution matrix [8], gap open cost of 16, gap extension cost of 2, and a short, spaced k-mer pattern for the prefilter stage (10111011) with six informative ('1') positions. Spaced k-mers are utilized in MMseqs2 to reduce the correlation between k-mers at neighboring positions, and to achieve better sensitivity and speed. The spaced k-mer pattern is chosen such that it is short in length in order to produce consecutive double k-mer matches (which are demanded by MMseqs2) within spacer fragments of 10-12 aa, and that the number of maximum overlapping informative positions is minimized.

Perfect or near-perfect hits (with no or 1-2 mismatches on the nucleotide level) are shown to be very reliable signals in predicting phage-host relationship and improve the taxonomic certainty of the prediction, even if there is only a single hit between a phage-host pair [4]. However, those hits are not well reflected in the pairwise P-value of the protein-level search. Therefore, all $q - t$ hits reported from the sensitive protein-level search will be aligned again on the nucleotide level with match reward of 1, mismatch penalty of 1, gap open cost of 10 and gap extension cost of 2. The protein-level search will compute a protein pairwise P-value (p_{prot}) for each hit and nucleotide alignment a nucleotide pairwise P-value (p_{nucl}). In order to prioritize near-perfect hits on the nucleotide level to gain precision without losing much sensitivity, we compute the pairwise P-value as

$$\exp\left(\min\left\{\left(0.5\log p_{\text{prot}} + 0.5\log p_{\text{nucl}}\right), \log p_{\text{nucl}}\right\}\right) \quad (1)$$

B. (2) Computing P-value of best hit

All hits of each q against the N_t proteins in a specific phage genome T are examined by their pairwise P-values, and the hit with the lowest pairwise P-value ("best hit") is retained. SpacePHARER computes the P-value of the best hit $p_{\text{bh}}(q)$ using first order statistics, i.e. the P-value of taking the minimum pairwise P-value ($p(q)$), given that a total of N_t pairwise P-values were examined:

$$p_{\text{bh}}(q) = P\left(p\left(q\right) \leq p\right) = 1 - \left(1 - p\right)^{N_t} \quad (2)$$

C. (3) Combining P-values using a modified truncated product method

In this step, we aim to combine the evidence from several best hits between a spacer set Q and a phage genome T. We sort the p_{bh} of the given set Q of N_q sequences in ascending order and denote the i'th p_{bh} as p_i. When combining independent P-values of individual hits, one needs to take into account the number of individual hits and the strength of each hit. The truncated product method combines independent P-values into a score by multiplying all $p_{\text{bh}}(q)$ smaller than a threshold p_0 [14],

$$S_{\text{comb}} = -\log \prod_{i=1}^{N_q} p_i^{I(p_i < p_0)}, \quad (3)$$

where $I(\cdot)$ is the indicator function that returns 1 if the argument is true and otherwise returns 0.

In SpacePHARER, we modified the truncated product method for better performance. We take the product of the smallest best-hit P-value p_1 times the ratio between

p_i and the threshold p_0 for all further p_i below the threshold p_0:

$$S_{\text{comb}} = -\log \left(p_1 \times \prod_{i=2}^{N_q} \left(\frac{p_i}{p_0} \right)^{I(p_i<p_0)} \right) \tag{4}$$

For the threshold, we set $p_0 = 1/(N_q + 1)$, which corresponds to marginal significance, with an E-value of $N_q/(N_q + 1)$ just below 1. This ensures that the combined score for null model-distributed P-values p_i only rarely gets boosted by a contribution from the second-best p_i.

D. (4) Determining true predictions

SpacePHARER predicts matches *de novo*, i.e. without relying on any known phage-host relationships, by controlling for estimated false discovery rate (FDR). The FDR is the proportion of false predictions among all predictions:

$$\text{FDR} = \frac{\text{FP}}{\text{FP} + \text{TP}} \tag{5}$$

We implemented an FDR estimation approach similar to that of the R package “fdrtool” [11]. In essence, we estimate the FDR by a Grenander decreasing density estimate of the empirical cumulative distribution function (ECDF). This non-parametric approach achieves its robustness by ensuring monotonicity of the FDR.

SpacePHARER uses a null model dataset to estimate the proportion of false predictions. The same search and statistical computation procedures described in Steps 1, 2 and 3 of the algorithm are performed on a given null model dataset, e.g. inverted phage ORFs or eukaryotic viral ORFs. Inverting target ORFs as null model dataset can be easily performed by specifying one parameter when preparing the input.

To compute an empirical P-value for each query spacer set Q, we sort for each Q the combined scores S_{comb} of matches in the original target dataset of phage proteomes in ascending order. For each S_{comb} value in the target dataset, we calculate an empirical P-value p_{emp} by using the fraction of $Q-T$ matches with a combined score that is below S_{comb} in the null model dataset. We denote the number of $Q-T$ matches below the cutoff as K and the total number of matches using the null model dataset as N_{null}. The empirical P-value is then computed as

$$p_{\text{emp}}(S_{\text{comb}}) = \frac{K + 0.5}{N_{\text{null}} + 1}, \tag{6}$$

where, to stabilize the estimate, we used half pseudocounts with P-values at 0 and 1. In the following, we abbreviate these empirical P-values as p, or p_Q for query set Q.

If we knew the fraction π_0 of false positives among all $Q-T$ matches, we could in principle estimate the false discovery rate simply as

$$\text{FDR}(p) = \frac{\text{FP}_p}{(\text{TP} + \text{FP})_p} \approx \frac{p\,\pi_0}{F_{\text{emp}}(p)}, \tag{7}$$

where $p\,\pi_0$ is the fraction of false positives with empirical P-value less than p_i. $F_{\text{emp}}(p)$ is the empirical cumulative distribution function of the p_Q, in other words $F_{\text{emp}}(p)$ is the number of query sets Q with best matches $p_Q \le p$.

We can increase the robustness of the estimate by using the fact that the true probability distribution of P-values $f(p)$ must be monotonously decreasing. This will also ensure that the FDR decreases with increasing p, which is often violated with the simple procedure above. The Grenander estimate [11] is a simple, efficient procedure to obtain a robust estimate $\hat{F}(p)$ of $F(p)$ from $F_{\text{emp}}(p)$ that has monotonously decreasing density $\hat{f}(p) = d\hat{F}(p)/dp$. We simply obtain the convex hull of the area under the $F_{\text{emp}}(p)$ curve, that is, the smallest function $\hat{F}(p)$ with $\hat{F}(p) \ge F_{\text{emp}}(p)$ that yields a convex area under the curve. This results in a piecewise constant, monotonously decreasing density function $\hat{f}(p) = d\hat{F}(p)/dp$ with steps at points p_i with $p_{\text{last}} = 1$. We estimate the proportion of true null hypotheses π_0 as the average density using the last two steps,

$$\pi_0 = \frac{\hat{F}(p_{\text{last}}) - \hat{F}(p_{\text{last}-2})}{p_{\text{last}} - p_{\text{last}-2}}. \tag{8}$$

Finally, we compute the estimated FDR corresponding to each empirical P-value p (Fig.1A) as

$$\text{FDR}(p) = \frac{\text{FP}_p}{(\text{TP} + \text{FP})_p} = \frac{p\,\pi_0}{\hat{F}(p)}. \tag{9}$$

By default, SpacePHARER has an FDR cutoff of 0.05, and reports all matches in the test whose S_{comb} corresponds to this FDR value or lower. Users can select other suitable FDR cutoffs to retain more or fewer predictions.

E. (5) Scanning for possible PAMs

For some CRISPR-Cas systems, protospacer adjacent motifs (PAMs) are required for the recognition of foreign invader sequences. After reporting phage-host pairs and their hits, SpacePHARER can perform a scan for possible PAMs. For this, SpacePHARER by default extracts 10 nt long fragments flanking the matched protospacer region at the 5’ and 3’ side, in guide-centric orientation (PAM is located on the strand that matches the spacer sequence). Users can increase or decrease the length of the flanking sequence. Both the 5’ and 3’ flanking sequences are searched in a list of consensus PAM patterns from representative CRISPR-Cas systems [5]. Since many CRISPR detection tools cannot reliably predict the orientation of the CRISPR array, the 5' and 3' flanking sequences on the reverse strand are also searched and two additional possible PAMs are reported. Users should refer to all possible PAMs without the accurate orientation information of the array.

II. OPTIMIZING PARAMETERS FOR SHORT FRAGMENTS SEARCH

Different substitution matrices are optimal for comparing sequences that have diverged to different degrees. By default, MMseqs2 search [10] uses the BLOSUM62 matrix with standard gap penalties: gap open cost of 11 and gap extend cost of 1 , which is more suited for long alignments and detecting weak protein similarities. Conversely for shorter sequences and higher protein similarity, one should consider a "shallower" (higher bit score per aligned column) matrix, and higher gap penalties to prevent gaps [9]. Searching with VTML40 matrix [8] with gap open cost of 16 and gap extend cost of 2 yielded the highest sensitivity with 20% on our test dataset at FDR cutoff of 0.05 (Figure S2). We introduced a series of VTML matrices in MMseqs2 to solve general problems of short sequence search. After introducing the additional nucleotide alignment step, the search parameter combination (VTML40 matrix, gap open cost of 16 and gap extend cost of 2) remains the highest in sensitivity (result not shown).

III. PREDICTING MATCHES USING BLASTN

We compared SpacePHARER's performance with the state-of-the-art method using BLASTN. To generate a comparable result, we performed the search step with BLASTN and the downstream FDR control with SpacePHARER. We used BLASTN [1] to first query the 80% test spacer dataset against 7,824 phage genomes, then against 7,824 inverted phage genomes or 11,304 eukaryotic viral genomes as a null model database. For all searches we used the parameters: -max_target_seqs 10000000 -dust no -word_size 7 -outfmt '6 std qcovs' and recorded the running time. Hits with at least 95% sequence identity and 95% query(spacer) coverage (i.e., one or two mismatches were allowed) were retained. We grouped the hits into matches (unique phage-host genome pairs) and retained the minimum pairwise E-value of the hits. We sorted the pairwise E-values of hits in ascending order for both searches and counted the matches at a given pairwise E-value cutoff. Therefore, we could calculate an FDR in the same way SpacePHARER does (described in section I.D) and compare the number of true predictions produced by the two methods (Figure 1B).

At FDR = 0.05, SpacePHARER predicted 2 and 1.5× more matches than BLASTN using 90% and 85% sequence identity and query coverage cutoffs (i.e allowing up to 4 and 6 mismatches, respectively) (Figure S2).

IV. HOST TAXONOMIC RANK ANALYSIS

To assess the sensitivity of SpacePHARER at different host taxonomic rank, we searched with CRISPR spacers extracted from 1,066 bacterial genomes against 809 phage genomes with annotated host taxonomy [4], then against inverted ORFs of the 809 phage genomes as null model dataset. For each phage, SpacePHARER predicted the host's lowest common ancestor (LCA) based on a weighted LCA procedure [7].

We demanded a stricter FDR cutoff of 0.02 for matches that should be taken into account for the host taxonomic rank prediction. In order to limit the number of false taxonomic predictions due to incomplete databases, the LCA result was further corrected according to the average nucleotide sequence identity of the reported matches [6]. We used the following cutoffs for maximal taxonomic resolution: > 86% (species), > 84% (genus), > 82% (family), > 80% (order), > 78% (class), > 76% (phylum), > 74% (kingdom). Lower values were assigned at the superkingdom level. The taxonomic FDR cutoff and sequence identity cutoffs are user-definable parameters for the weighted LCA procedure.

We searched with the above-mentioned spacer dataset against phage genomes using BLASTN with parameters: blastn-short -dust no -word_size 7 -outfmt '6 std qcovs' -evalue 1 -gapopen 10 -gapextend 2 -penalty -1 [4]. Hits with at least 95% sequence identity and 95% query(spacer) coverage were retained (i.e., one or two mismatches were allowed). For each phage, the bacterium with the lowest pairwise E-value was predicted to be its host. Note that in Edwards et al., the authors searched with the phage genomes against the spacer dataset, and demanded 100% spacer coverage.

For ranks lower than phylum, we only included the predictions with the taxonomic resolution of the respective rank or below. At the species level, SpacePHARER predicted 142/237 hosts (60%), compared to 112/232 hosts of BLASTN (48%). SpacePHARER predicted the correct host for more phages at all taxonomic ranks, while including most of the BLASTN predictions on the same rank and sometimes even those agreeing only on a higher rank(Figure 1C, Figure S3).

Incomplete reference databases remain an issue for phage-host relationship predictions. To simulate scenarios where the database is very incomplete, we progressively exclude 25% and 50% of the host genomes in the spacer dataset, and compare the performance between BLASTN and SpacePHARER. SpacePHARER predicted the correct host for more phages than BLASTN at all taxonomic ranks when we searched with 50% and 75% of original host spacer dataset (Figure S4).

V. IDENTIFYING MIS-ANNOTATIONS IN EUKARYOTIC VIRAL DATASET

Throughout this study we used the set of *eukaryotic viral genomes* as a null model dataset, assuming any match between a prokaryotic genome and a eukaryotic virus is false. Here, we used SpacePHARER's second mode of FDR control to detect viruses that were potentially mis-annotated as eukaryotic viruses. To that end, we first ran the SpacePHARER workflow with the full spacer dataset against the *eukaryotic viral genomes* as the target database, and then, against inverted *eukaryotic viral* ORFs as the null model database. We used the null set to estimate the FDR as described in section I.D.

By applying the same FDR cutoff of 0.05, we identified 11 viruses out of the 11,304 that matched a prokaryotic host (yielding a total of 12 matches). We observed three groups within these matches. The first group consisted of two matches between the smacovirus family (KP264966.1 and KY086299.1) and the archaeon CP005934.1 (*Candidatus Methanomassiliicoccus intestinalis*). Indeed this family has been recently reported as mis-annotated as "eukaryotic virus" by Díez-Villaseñor and Rodriguez-Valera [3]. The second group consisted of two matches between KT809302.1 (Haloarcula californiae icosahedral virus 1) and family *Halobacteriaceae* (CP001687.1 and LIST01000008.1). These matches are likely due to mis-annotation of the virus as "eukaryotic virus". The labeled host of this virus is *Haloarcula californiae*, which is an archaeon that belongs to the same family as our matches. The third group consisted of 8 members of the genus *Mimivirus* that were matched to HE978663.1 (*Ruminococcus sp.* JC304) and JAAF01000022.1 (*Fusobacterium necrophorum* DAB). Table I shows the standard output from SpacePHARER of this search. We suspect the matches of the third group are due to spacer mis-annotation and do not represent a real virus-host relationship. It was previously reported that Mimiviruses acquire bacterial genes, even of the class *Clostridia* [12][13]. In the case of *Ruminococcus sp.* JC304, when we inspected the bacterial genomic region from which the spacers were extracted, we found that the entire region is likely to be a full bacterial ORF, rather than a CRISPR array. Thus, we conclude that in these cases, the mis-annotation is of the CRISPR array, rather than of the virus.

VI. SOFTWARE VERSIONS

Name	Version
SpacePHARER	Git: 1d1f1b2
BLASTN	2.9.0+

TABLE II. Software versions used in this manuscript.

[1] Altschul, S.F. et al (1990). Basic local alignment search tool. *J. Mol. Biol.*, **215**(3), 403–410.

[2] Brunson, J.C. (2020). ggalluvial: Layered grammar for alluvial plots. *J. Open Source Softw.*, **5**(49), 2017.

[3] Díez-Villaseñor, C. and Rodriguez-Valera, F. (2019). CRISPR analysis suggests that small circular single-stranded dna smacoviruses infect archaea instead of humans. *Nat. Commun.*, **10**(1), 294.

[4] Edwards, R.A. et al (2015). Computational approaches to predict bacteriophage–host relationships. *FEMS Microbiol. Rev.*, **40**(2), 258–272.

[5] Leenay, R.T. and Beisel, C.L. (2017). Deciphering, communicating, and engineering the crispr pam. *Journal of molecular biology*, **429**(2), 177–191.

[6] Levy Karin, E. et al (2020). Metaeuk—sensitive, high-throughput gene discovery, and annotation for large-scale eukaryotic metagenomics. *Microbiome*, **8**(1), 48.

[7] Mirdita, M. et al (2021). Fast and sensitive taxonomic assignment to metagenomic contigs. *Bioinformatics.* btab184.

[8] Müller, T. et al (2002). Estimating amino acid substitution models: A comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol. Biol. Evol.*, **19**(1), 8–13.

[9] Pearson, W.R. (2013). Selecting the Right Similarity-Scoring Matrix. *Current Protocols in Bioinformatics*, **43**(1), 3.5.1–3.5.9.

[10] Steinegger, M. and Söding, J. (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**(11), 1026–1028.

[11] Strimmer, K. (2008). A unified approach to false discovery rate estimation. *BMC Bioinformatics*, **9**(1), 303.

[12] Yoshida, T. et al (2011). Mimivirus reveals mre11/rad50 fusion proteins with a sporadic distribution in eukaryotes, bacteria, viruses and plasmids. *Virology journal*, **8**, 427–427.

[13] Yutin, N. et al (2014). Origin of giant viruses from smaller dna viruses not from a fourth domain of cellular life. *Virology*, **466-467**, 38 – 52. Special issue: Giant Viruses.

[14] Zaykin, D. et al (2002). Truncated product method for combining p-values. *Genet. Epidemiol.*, **22**(2), 170–185.

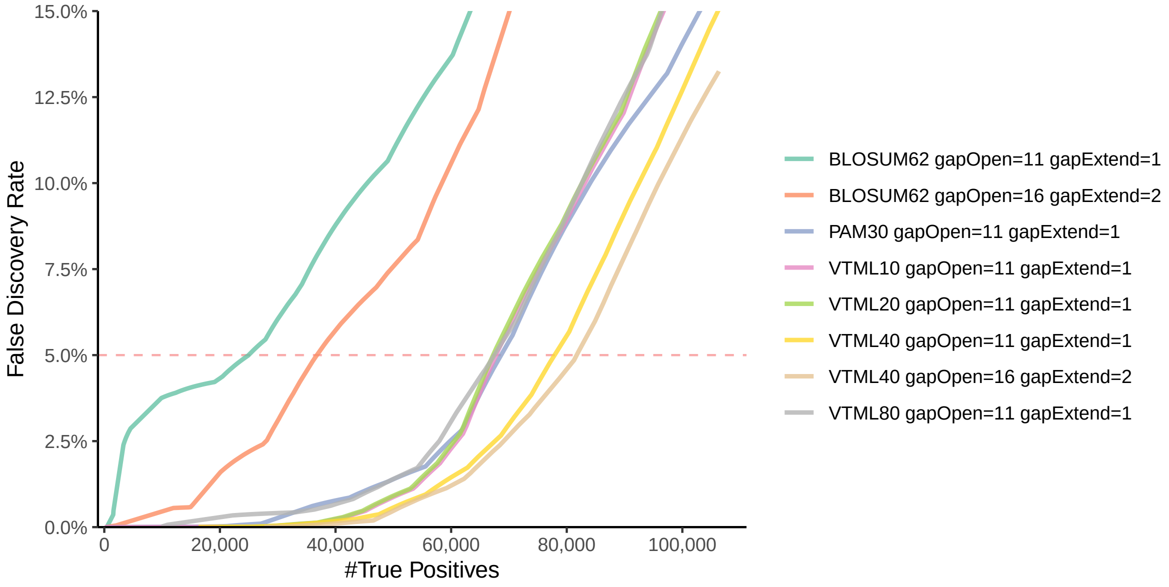


FIG. 1. Performance comparison of SpacePHARER with different search parameters (substitution matrix and gap penalties), evaluated by the number of true positive (TP) predictions at different false discovery rates (FDRs). Predictions were made by using an optimization spacer dataset (6,067 genomes, 20% of all prokaryotic genomes) against a database of 7,824 phage genomes, with inverted phage ORFs as null model database. Searching with VTML40 matrix with gap open (16) and gap extend (2), among various combinations of substitution matrix and gap penalties, yields more true positive matches than any other parameter combination at FDR cutoff of 0.05.

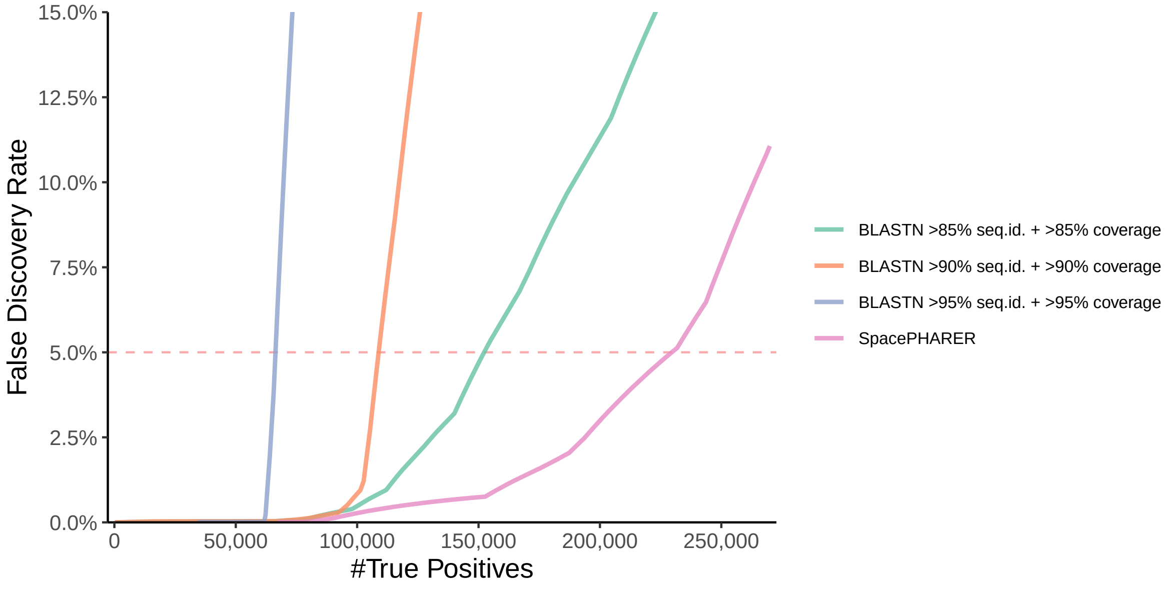


FIG. 2. Performance comparison of SpacePHARER with BLASTN using different sequence identity and query coverage cutoffs (95%, 90% and 85%), evaluated by the number of true positive (TP) predictions at different false discovery rates (FDRs). Predictions were made by using a spacer test dataset (24,322 genomes, 80% of all prokaryotic genomes) against a database of 7,824 phage genomes, with inverted phage ORFs as null model database. (Note that the FDR control procedure developed for SpacePHARER is not standard for BLASTN and has been applied here only for the purpose of FDR analysis.)

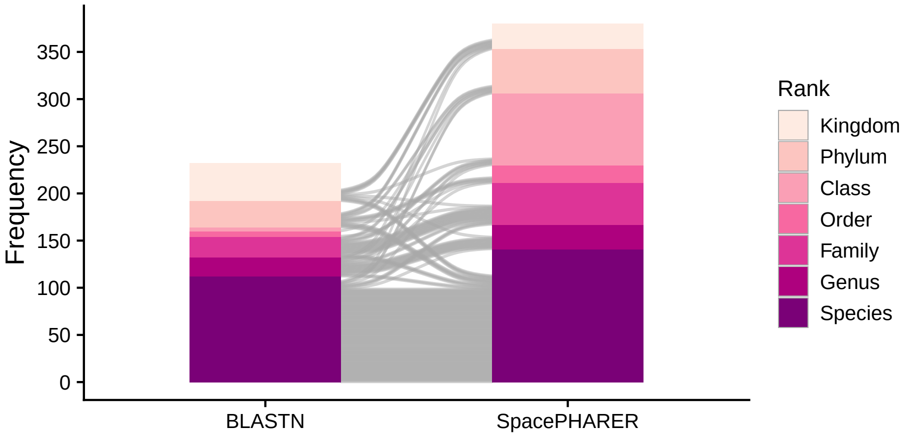


FIG. 3. Performance comparison of BLASTN (left) and SpacePHARER (right), evaluated by the number of host predictions that agree with annotated host taxonomy at different taxonomic ranks. The grey alluvia [2] represent the host predictions that were made by both SpacePHARER and BLASTN. Predictions were made using a validation spacer dataset (1,066 genomes) against a validation database of 809 phage genomes with annotated host taxonomy. SpacePHARER prediction was further corrected with inverted phage ORFs as null model database, and FDR cutoff of 0.02.

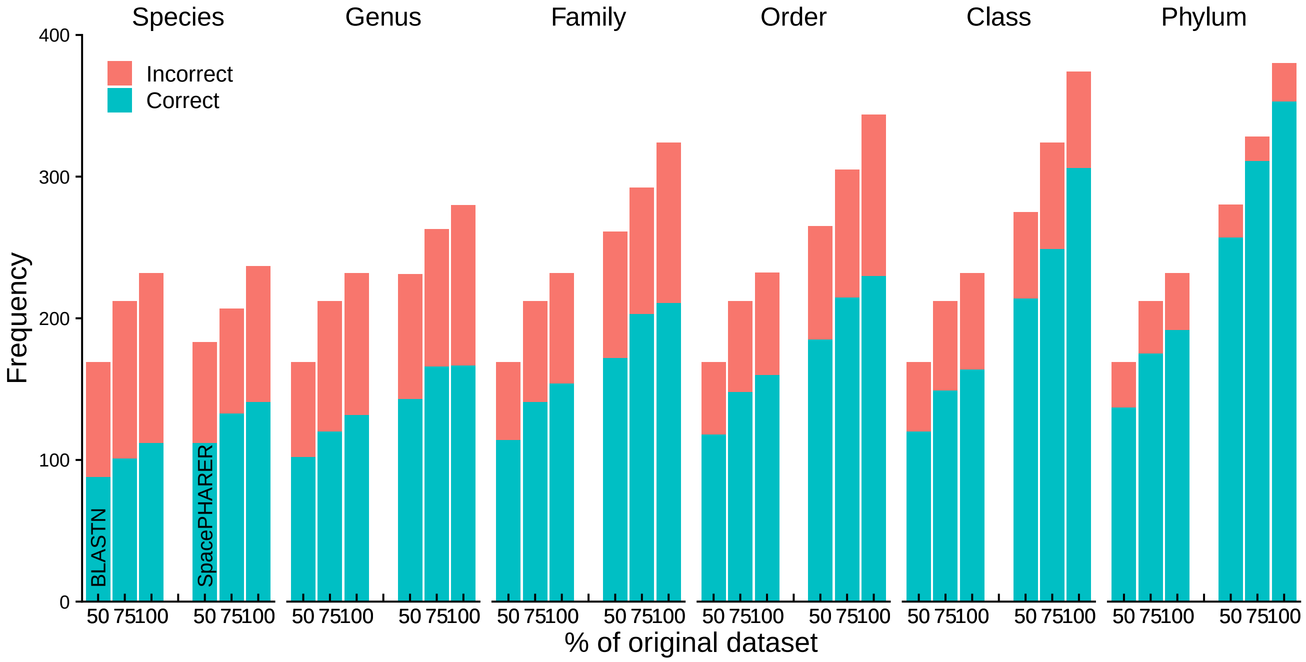


FIG. 4. Performance comparison of BLASTN (left) and SpacePHARER (right) as for Figure 1C, but on incomplete databases. The host spacer dataset was progressively depleted from 100% of genomes (1,066) to 75% (800) and 50% (533). Performance is evaluated by the number of host predictions that agree with annotated host taxonomy at different taxonomic ranks.

```
#CP005934.fas                                     KP264966.1  7.588E+01  6
>CP005934.1_930280_937725_19_spacer_931524_35     KP264966.1  1.023E-04  35  3   546     578     CCT|-             -|AGG
>CP005934.1_930280_937725_20_spacer_931590_37     KP264966.1  2.833E-04  1   36  2241    2206    CCT|-             -|AGG
>CP005934.1_930280_937725_23_spacer_931792_37     KP264966.1  1.034E-09  1   36  1821    1786    CCA|-             -|TGG
>CP005934.1_930280_937725_24_spacer_931860_37     KP264966.1  3.121E-07  3   35  606     638     CCA|TGG           -|TGG
>CP005934.1_930280_937725_25_spacer_931928_36     KP264966.1  3.399E-13  1   36  2161    2126    CCG|-             -|CGG
>CP005934.1_930280_937725_25_spacer_931928_36     KP264966.1  1.713E-11  2   34  2160    2128    CCG|-             -|CGG
#CP005934.fas                                     KY086299.1  5.640E+01  4
>CP005934.1_930280_937725_19_spacer_931524_35     KY086299.1  6.205E-04  35  3   1922    1890    CCT|-             -|AGG
>CP005934.1_930280_937725_23_spacer_931792_37     KY086299.1  3.399E-13  2   37  641     676     CCA|-             -|AGG
>CP005934.1_930280_937725_20_spacer_931590_37     KY086299.1  4.613E-05  1   36  220     255     CCT|-             -|TGG
>CP005934.1_930280_937725_23_spacer_931792_37     KY086299.1  3.399E-13  1   36  640     675     CCA|-             -|TGG
#LIST01000008.fas                                 KT809302.1  1.295E+01  1
>LIST01000008.1_120573_126312_45_spacer_123484_36 KT809302.1  2.376E-06  36  1   22375   22340   -|-               TTC|-
#CP001687.fas                                     KT809302.1  2.639E+01  2
>CP001687.1_1415738_1419119_25_spacer_1417344_34  KT809302.1  1.137E-07  6   32  6826    6852    -|CAAGAA          -|ACGGGATT
>CP001687.1_1415738_1419119_25_spacer_1417344_34  KT809302.1  1.137E-07  32  6   6852    6826    -|CAAGAA          -|ACGGGATT
#HE978663.fas                                     JN258408.1  1.054E+02  2
>HE978663.1_7481_7851_2_spacer_7588_70            JN258408.1  1.755E-28  2   70  806538  806606  TTC|-             -|-
>HE978663.1_7481_7851_4_spacer_7765_58            JN258408.1  5.707E-20  2   58  806715  806771  TTC|-             -|-
#HE978663.fas                                     JX885207.1  9.775E+01  2
>HE978663.1_7481_7851_2_spacer_7588_70            JX885207.1  1.755E-28  2   70  767273  767341  TTC|-             -|-
>HE978663.1_7481_7851_4_spacer_7765_58            JX885207.1  1.187E-16  2   58  767450  767506  TTC|-             -|-
#HE978663.fas                                     KF527229.1  8.186E+01  2
>HE978663.1_7481_7851_1_spacer_7510_49            KF527229.1  2.905E-18  2   49  935992  935945  TTC|-             -|-
>HE978663.1_7481_7851_4_spacer_7765_58            KF527229.1  5.707E-20  2   58  935836  935780  TTC|-             -|-
#HE978663.fas                                     KU877344.1  9.775E+01  2
>HE978663.1_7481_7851_2_spacer_7588_70            KU877344.1  1.755E-28  2   70  780352  780420  TTC|-             -|-
>HE978663.1_7481_7851_4_spacer_7765_58            KU877344.1  1.187E-16  2   58  780529  780585  TTC|-             -|-
#HE978663.fas                                     JX975216.1  1.015E+02  2
>HE978663.1_7481_7851_2_spacer_7588_70            JX975216.1  1.755E-28  2   70  781866  781934  TTC|-             -|-
>HE978663.1_7481_7851_4_spacer_7765_58            JX975216.1  2.682E-18  2   58  782043  782099  TTC|-             -|-
#HE978663.fas                                     MG779360.1  1.093E+02  2
>HE978663.1_7481_7851_2_spacer_7588_70            MG779360.1  3.447E-30  2   70  9786    9854    -|-               -|-
>HE978663.1_7481_7851_4_spacer_7765_58            MG779360.1  5.707E-20  2   58  9963    10019   -|-               -|-
#HE978663.fas                                     JN885991.1  4.536E+01  2
>HE978663.1_7481_7851_3_spacer_7687_49            JN885991.1  2.061E-02  2   46  497977  498021  CCT|-             TTG|AGG
>HE978663.1_7481_7851_4_spacer_7765_58            JN885991.1  5.707E-20  2   58  498055  498111  -|-               -|-
#JAAF01000022.fas                                 KY684109.1  1.295E+01  1
>JAAF01000022.1_41_3914_26_spacer_1726_36         KY684109.1  2.376E-06  1   36  185359  185324  TCT|TGAAGTTT      TCA|-
```

TABLE I. Sample output format of SpacePHARER, demonstrated by matches when searching the full spacer dataset against eukaryotic viral ORFs as a target database and inverted eukaryotic viral ORFs as null model database. Each match line starts with '#', followed by the prokaryote accession (the file from which spacers were extracted), viral genome accession, S_{comb} and the number of hits in the match. Each hit line starts with '>', followed by the spacer sequence header, viral genome accession, p_{bh}, spacer start, spacer end, viral genome start, viral genome end, and the possible PAM sequences on forward and reverse strand (5'|3'). Additionally (not shown), the aligned sequences can be printed following each hit line.

6 Minor contributions

6.1 Protein Sequence Analysis Using the MPI Bioinformatics Toolkit

Publication:

Protein Sequence Analysis Using the MPI Bioinformatics Toolkit

F. Gabler, S.Z. Nam, S. Till, **M. Mirdita**, M. Steinegger, J. Söding, A.N. Lupas, V. Alva†

(†) corresponding author

Current Protocols in Bioinformatics (2020), 72, e108.
Cited 84 times since 12/2020.

Manuscript abstract

The MPI Bioinformatics Toolkit (`toolkit.tuebingen.mpg.de`) provides interactive access to a wide range of the best-performing bioinformatics tools and databases, including the state-of-the-art protein sequence comparison methods HHblits and HHpred. The Toolkit currently includes 35 external and in-house tools, covering functionalities such as sequence similarity searching, prediction of sequence features, and sequence classification. Due to this breadth of functionality, the tight interconnection of its constituent tools, and its ease of use, the Toolkit has become an important resource for biomedical research and for teaching protein sequence analysis to students in the life sciences. In this article, we provide detailed information on utilizing the three most widely accessed tools within the Toolkit: HHpred for the detection of homologs, HHpred in conjunction with MODELLER for structure prediction and homology modeling, and CLANS for the visualization of relationships in large sequence datasets.

Author contributions

F.G.: Software; writing-review & editing. S.Z.N.: Software. S.T.: Software. **M.M.**: Software. M.S.: Software. J.S.: Software. A.L.: Funding acquisition; writing-review & editing. V.A.: Conceptualization; project administration; software; supervision; visualization; writing-original draft; writing-review & editing.

6.2 PredictProtein – Predicting Protein Structure and Function for 29 Years

Publication:

PredictProtein – Predicting Protein Structure and Function for 29 Years

M. Bernhofer, C. Dallago[†], T. Karl, V. Satagopam, M. Heinzinger, M. Littmann, T. Olenyi, J. Qiu, K. Schütze, G. Yachdav, H. Ashkenazy, N. Ben-Tal, Y. Bromberg, T. Goldberg, L. Kajan, S. O'Donoghue, C. Sander, A. Schafferhans, A. Schlessinger, G. Vriend, **M. Mirdita**, P. Gawron, W. Gu, Y. Jarosz, C. Trefois, M. Steinegger, R. Schneider, B. Rost

(†) corresponding author

Nucleic Acids Research (2021), 49, W535-W540.

Cited 15 times since 5/2021.

Manuscript abstract

Since 1992 PredictProtein (`predictprotein.org`) is a one-stop online resource for protein sequence analysis with its main site hosted at the Luxembourg Centre for Systems Biomedicine (LCSB) and queried monthly by over 3,000 users in 2020. PredictProtein was the first Internet server for protein predictions. It pioneered combining evolutionary information and machine learning. Given a protein sequence as input, the server outputs multiple sequence alignments, predictions of protein structure in 1D and 2D (secondary structure, solvent accessibility, transmembrane segments, disordered regions, protein flexibility, and disulfide bridges) and predictions of protein function (functional effects of sequence variation or point mutations, Gene Ontology (GO) terms, subcellular localization, and protein-, RNA-, and DNA binding). PredictProtein's infrastructure has moved to the LCSB increasing throughput; the use of MMseqs2 sequence search reduced runtime five-fold; user interface elements improved usability, and new prediction methods were added. PredictProtein recently included predictions from deep learning embeddings (GO and secondary structure) and a method for the prediction of proteins and residues binding DNA, RNA, or other proteins. PredictProtein.org aspires to provide reliable predictions to computational and experimental biologists alike. All scripts and methods are freely available for offline execution in high-throughput settings.

Selected contributions

M.M. implemented and maintains a server to compute MSAs for PredictProtein.

6.3 Going to extremes - a metagenomic journey into the dark matter of life

Publication:

Going to extremes - a metagenomic journey into the dark matter of life

A. Aevarsson[†], A. Kaczorowska, B.T. Adalsteinsson, J. Ahlqvist, S. Al-Karadaghi, J. Altenbuchner, H. Arsin, Ú. Áugúst Átlasson, D. Brandt, M. Cichowicz-Cieślak, K. A S Cornish, J. Courtin, S. Dabrowski, H. Dahle, S. Djeffane, S. Dorawa, J. Dusaucy, F. Enault, A. Fedøy, S. Freitag-Pohl, O.H. Fridjonsson, C. Galiez, E. Glomsaker, M. Guérin, S.E. Gundesø, E.E. Gudmundsdóttir, H. Gudmundsson, M. Håkansson, C. Henke, A. Helleux, J.R. Henriksen, S. Hjörleifdóttir, G.O. Hreggvidsson, A. Jasilionis, A. Jochheim, I. Jónsdóttir, L.B. Jónsdóttir, A. Jurczak-Kurek, T. Kaczorowski, J. Kalinowski, L.P. Kozlowski, M. Krupovic, K. Kwiatkowska-Semrau, O. Lanes, J. Lange, J. Lebrat, J. Linares-Pastén, Y. Liu, S.A. Lorentsen, T. Lutterman, T. Mas, W. Merré, **M. Mirdita**, A. Morzywołek, E.O. Ndela, E. Nordberg Karlsson, E. Olgudóttir, C. Pedersen, F. Perler, S.K. Pétursdóttir, M. Plotka, E. Pohl, D. Prangishvili, J.L. Ray, B. Reynisson, T. Róbertsdóttir, R. Sandaa, A. Sczyrba, S. Skírnisdóttir, J. Söding, T. Solstad, I.H. Steen, S.K. Stefánsson, M. Steinegger, K. Stange Overå, B. Striberny, A. Svensson, M. Szadkowska, E.J. Tarrant, P. Terzian, M. Tourigny, T. van den Bergh, J. Vanhalst, J. Vincent, B. Vroling, B. Walse, L. Wang, H. Watzlawick, M. Welin, O. Werbowy, E. Wons, R. Zhang

(†) corresponding author

FEMS Microbiology Letters (2021), 368(12), fnab067.
Cited 4 times since 6/2021.

Manuscript abstract

The Virus-X—Viral Metagenomics for Innovation Value—project was a scientific expedition to explore and exploit uncharted territory of genetic diversity in extreme natural environments such as geothermal hot springs and deep-sea ocean ecosystems. Specifically, the project was set to analyse and exploit viral metagenomes with the ultimate goal of developing new gene products with high innovation value for applications in biotechnology, pharmaceutical, medical, and the life science sectors. Viral gene pool analysis is also essential to obtain fundamental insight into ecosystem dynamics and to investigate how viruses influence the evolution of microbes and multicellular organisms. The Virus-X Consortium, established in 2016, included experts from eight European countries. The unique approach based on high throughput bioinformatics technologies combined with structural and functional studies resulted in the development of a biodiscovery pipeline of significant capacity and scale. The activities within the Virus-X consortium cover the entire range from bioprospecting and methods development in bioinformatics to protein production and characterisation, with the final goal of translating our results into new products for the bioeconomy. The significant impact the consortium made in all of these areas was possible due to the successful cooperation between expert teams that worked together to solve a complex scientific problem using state-of-the-art technologies as well as developing novel tools to explore the virosphere, widely considered as the last great frontier of life.

Selected contributions

M.M. implemented software to track the status of each target through the work packages, contributed to method development in Platform 2 and reviewed the manuscript.

7 Discussion and outlook

The realization that protein sequence similarity often stems from evolutionary relationships (gene duplications and species divergence) was revolutionary for our understanding of biology and evolution. Even more critical was the observation that very remotely related protein sequences often have conserved structures and, above certain thresholds, conserved functions. This allowed transfer of functional annotation from experimentally studied model species to other organisms and recently helped unlock the prediction of protein structures at an accuracy nearly indistinguishable from crystal structures. Our ever-expanding repositories of known sequences allow novel phylogenetic and evolutionary insights every time the genome of a new organism is explored.

The progress of sequencing technologies over the last forty years has led to an exponential growth in the amount of available sequences. While this treasure trove of data allowed more powerful analyses than ever before, the sheer volume of data made the development of efficient homology search methods critical. The advent of metagenomics put these challenges at the forefront.

In this work I have presented three computational methods I developed and published during the course of my doctoral studies. These improve upon the state-of-the-art for homology search, taxonomic assignment and protein structure prediction. The methods are user-friendly and accessible for a wide range of users.

One of the most straightforward methods to exploit metagenomics data is their *in silico* analysis to identify proteins with commercially desirable properties [Robinson et al., 2021]. As participants of the Virus-X consortium (Aevarsson et al. [2021]; see section 6.3) we helped build a pipeline for discovering novel thermostable proteins from extremophilic environments. This was possible because of the integration of MMseqs2 (and many other methods) to filter down the large metagenomic samples to smaller sets that could be extensively studied.

Apart from identifying individual proteins with a specific function, metagenomics data have enriched existing protein databases, providing us with a more complete picture of the true diversity of naturally evolved sequences. Metagenomic-based protein databases like MGnify [Mitchell et al., 2020] and BFD [Jumper et al., 2021a] contain several billion protein sequences. The projects presented in this work contributed three databases constructed from metagenomic data. First, the SRC/MERC databases that were released as part of the Plass [Steinegger et al., 2019b] manuscript contain hundreds of million to billions of sequences assembled from soil and marine metagenomes and metatranscriptomes. Second, we released several million likely eukaryotic proteins from marine metagenomes enriched for protists and an accompanying database of tens of millions of protein profiles as part of MetaEuk [Levy Karin et al., 2020]. The ColabFoldDB that was released as part of ColabFold [Mirdita et al., 2021] contains over 200 million protein profiles. A part of the success of AlphaFold2 stems from exploiting the newly found sequence diversity to build more diverse MSAs, informing the most accurate to-date prediction of protein structures.

An important and rarely discussed aspect of computational methods is their impact on climate change. Already today, data centers and high-performance computing are substantial contributors to climate warming emissions [Lannelongue et al., 2021]. Our methods perform analyses much faster than their state-of-the-art competitors at similar or better quality, thus reaching the same or better results at a significantly reduced carbon footprint. In the MMseqs2 app manuscript we show that MMseqs2 can perform profile searches hundreds to thousands of times faster than its competitors. MMseqs2 taxonomy is between 2× and 18× faster than the state-of-the-art taxonomy assignment method CAT. ColabFold's MSA generation stage is 20-30× faster than the AlphaFold2's. SpacePHARER is 47× faster than a BLASTN based approach.

The performance goals that we set for the MMseqs2 software and the diverse set of methods we have built on top resulted in a complex code base and came with a corresponding maintenance burden. To aid with development and facilitate maintenance I implemented continuous integration: before any change is made to the code base we run an extensive test suite across many hardware-architectures, compilers and operating systems. We provide a detailed documentation of MMseqs2 in our extensive user guide (see Appendix A1) and have organized workshops to train the community in applying MMseqs2 and incorporate their feedback. The modular nature of our software and the extensive documentation have also facilitated the participation of multiple external contributors.

We committed to the principles of open science early on. MMseqs2 and all software that we built on top is open-source and comes with free and open source licenses. We make all software packages available on popular repositories such as Bioconda [Grüning et al., 2018], Homebrew (`brew.sh`) or pip (`pypi.org`). Additionally, we closely follow and respond to the valuable feedback provided by our large user base. Finally, we established a support network to remain in contact with our users to quickly resolve issues, often turning discussions about the use of MMseqs2 into fruitful collaborations.

In the following I would like to focus on some of the challenges, limitations and upcoming work related to the three main projects presented in this work.

7.1 MMseqs2 App and Server

I developed the MMseqs2 app and server to make MMseqs2 user-friendly and accessible. Beyond its original purpose, the MMseqs2 web server has been used for performing additional tasks. Specifically, we included the eukaryotic proteins found by MetaEuk as a searchable database in the server and PredictProtein and ColabFold use it for constructing diverse MSAs.

We plan to expose further MMseqs2 functionality within the web server. FoldSeek (in preparation) is an upcoming algorithm for extremely fast structure-to-structure search. As millions of predicted high-quality protein structures will soon be available [Tunyasuvunakool et al., 2021, Callaway, 2021, Varadi et al., 2021], it is important to have algorithms ready to process these data. Our approach reduces three-dimensional structures to a conformational state alphabet, which can efficiently be queried with algorithms developed for sequence searches. Moreover, we plan additional extensions to our web server to include MMseqs2 taxonomic annotation capabilities, to make these easier to use.

The MMseqs2 app is geared towards single-sequence searches. However, the enormous size of databases constructed from metagenomics poses a challenge to fast searches with individual sequences – especially as these protein databases are now growing into billions of sequences. The main issue lies with the prefiltering algorithm, which uses data structures that consume RAM proportionately to the cumulative length of the sequences in the target database. In non-server MMseqs2 batch searches, the data structures can be built and queried in separate chunks and the construction costs can be amortized across all queries. Thus, memory consumption is less of a concern there. However, processing queries within milliseconds to seconds with the MMseqs2 server, the search database and its required data structures have to fully reside in system memory. Therefore, memory requirements might become a major cost factor, especially as the reduction of RAM cost per gigabyte has not kept up with Moore's Law [McCallum, 2021].

Faced with a similar issue in ColabFold, we used a clustered homology search approach. We first search with the normal MMseqs2 search workflow against the cluster consensus sequences and then, for each hit, expand the search result with the cluster members. This way, system memory only needs to hold data structures for the prefiltering against the cluster centroids, reducing the memory footprint drastically.

Another solution we are exploring is to build a novel search algorithm that exploits the high read-throughput of modern storage hardware. Currently, a typical consumer NVMe storage device offers 2 TB of storage space and up to 7 GB/s read throughput for around 300$. Martin Steinegger, Johannes Söding and I are currently developing an algorithm and software that allows searching through 40TB of data in under 5 minutes on a built-to-purpose server filled with twenty such devices. We call this algorithm PetaSearch (in preparation) and plan to offer a web server based on the MMseqs2 server to allow searching through all proteins we can extract from the SRA.

7.2 MMseqs2 Taxonomy

Introducing taxonomic assignment capabilities to MMseqs2 has contributed to several other projects, such as SpacePHARER and ColabFold. SpacePHARER annotates either phages or their bacterial hosts, if given taxonomic labels for the other. ColabFold uses taxonomic labels for MSA pairing (see below).

To help users interpret their results, I implemented two output modes to allow visual inspection of the taxonomy results. The first mode is a Kraken-compatible format that can be used to visualize the results with tools like Pavian [Breitwieser and Salzberg, 2020]. The other output for visual inspection is an interactive Krona HTML file [Ondov et al., 2011]. We plan on making MMseqs2 taxonomy easier to use, by supporting more output formats and by providing it as a web server in the future.

A limitation of MMseqs2 taxonomy is that it uses reference-based assignment of taxonomic labels. This means it is sensitive to the quality of the reference databases, which have been shown to suffer from contamination [Steinegger and Salzberg, 2020] and uneven sampling. Alternative taxonomic databases, such as the GTDB are also limited by the (metagenome assembled) genomes available. However, the community is continuously improving the reference databases and MMseqs2 taxonomy will be able to assign better labels using higher quality databases. Furthermore, MMseqs2 taxonomy is well suited to deal with the increasing size of the databases.

In recent years, long read sequencing technology has drastically improved in read length and base calling accuracy. Even single reads of mega base length are now occasionally produced by modern sequencing methods (e.g., the 1.2 and 2.3 megabase long reads found in Payne et al. [2019]). MMseqs2 taxonomy should be well suited for annotating long sequences. We hope to repeat the benchmarking on long read data in the future and optimize MMseqs2 taxonomy to long read input.

7.3 ColabFold

ColabFold has become a widely used method in the short time that has passed since its release. Its MSA server has processed over 700 000 MSAs for users and is averaging ∼10 000 MSAs a day.

In its first release, ColabFold included a component to model protein complexes. Since then, AlphaFold2 has been specifically retrained on protein complexes and released as an updated version called AlphaFold-multimer [Evans et al., 2021]. By testing it on a small benchmark, we found AlphaFold2-multimer outperforms ColabFold's complex modeler. We thus integrated AlphaFold2-multimer based modeling in ColabFold.

High-quality protein complex predictions require a paired MSA. In a paired MSA, directly related protein sequences, e.g., those which share the same species - are placed in the MSA line and are concatenated. As UniRef contains taxonomic information, pairing sequences from corresponding taxa is straightforward. However, ColabFold's environmental databases do not currently contain any metadata and thus can only be used as unpaired sequence features. To further improve complex modeling, we plan on gathering available taxonomic metadata for metagenomic sequences and assigning taxonomic labels for sequences without metadata.

We plan on further extending ColabFoldDB with additional metagenomic and -transcriptomic data sets. Together with other planned improvements in MMseqs2's sensitivity we want to investigate, we hope that ColabFold will be able to find a sufficient number of homologs to ensure high quality protein structure predictions for most sequences.

8 Conclusion

The projects I worked on during my doctoral studies and continue to develop have contributed to the community's ability to investigate and explore billions of metagenomic sequences from various environments. With over 500 citations in total, it is clear these tools can be (and already have been) put to use by various research groups and overcome some of the main challenges in conducting metagenomic research (e.g., large database sizes, complex assembly tasks, functional and taxonomic annotation and more). The focus on computational efficiency has enabled many large-scale analyses and made replacing less efficient methods possible.

In particular, AlphaFold2 relies on methods that I have contributed, developed, maintained and for which I have provided user-support during my doctoral research: The tools HH-suite, MMseqs2 and Plass and the databases Uniclust and PDB70. During its self-distillation training, AlphaFold2 uses $\sim 350\,000$ diverse clusters that were selected (and enriched by HHblits) from the Uniclust30 database. During inference its homology search and template detection relies on HHblits/HHsearch searches against the Uniclust30 and PDB70. Additionally, it searches through the BFD – a database that primarily contains sequences assembled by Plass and clustered by MMseqs2.

As I conclude the presentation of my doctoral projects, I would like to focus on a few particularly exciting technological developments that lie ahead in the next decade. These offer many new opportunities to develop novel algorithms and provide benefits of sequence-based analyses to new fields.

Long read sequencing technologies have made large strides in the last years. Particularly, the sequencing technology company Oxford Nanopore Technologies stands out. I already mentioned ultra-long reads and protein sequencing as tantalizing applications of this sequencing platform. The continuing miniaturization efforts (Flongle, MinION Mk1c) offer a glimpse into a future, where an end-to-end integrated platform might be available that deals with all steps from DNA extraction to base-calling and bioinformatic analysis on a single chip [Wang et al., 2021]. Such a small, affordable device could be easily deployed anywhere e.g., for passive pathogen monitoring. Already now, wastewater monitoring for pathogens has proven itself during the ongoing COVID-19 pandemic as an early indicator for soon-to-be spiking infection numbers [Karthikeyan et al., 2021].

For the compute hardware side, I want to briefly focus on two areas of upcoming technologies that seem particularly exciting for novel algorithms for homology searches:

On-CPU caches will soon exceed the gigabyte threshold (AMD Epyc 7773X now contains 804MB of on-chip caches). Thus, large data structures could be held directly in the CPU caches to accelerate memory-bound algorithms. The MMseqs2 prefilter already implements many cache-optimizations and would be a good target to see re-imagined in the future. Further reducing the relatively slow random accesses to RAM in the MMseqs2 prefilter would yield large benefits for most methods presented here.

Solid-state disks and particularly such devices based on the NVMe standard offer high storage density and high-throughput read/write operations. The price per terabyte of solid-state storage is projected to drop below traditional hard disk prices in the mid 2020s [Floyer, 2021]. Multiple NVMe-devices in-tandem will soon offer aggregated read bandwidths of hundreds of gigabytes per second. Additionally, relatively powerful CPUs are now found on every NVMe device. Consumer NVMe storage devices containing multi-core CPUs with clock speeds above 1 GHz and multiple gigabytes of RAM are now common. Efforts are underway to give programmatic access to these to offload data-processing directly onto the storage medium. This would offer an additional opportunity to accelerate MMseqs2 and PetaSearch (see section 7.1).

Novel algorithms will be needed to exploit the capabilities of upcoming technologies for the analysis of the data produced by biological (and especially metagenomic) experiments.

ColabFold and future structure predictors, might start a new era of bioinformatics with structure-analysis at its core. Further accelerating these methods will allow us to predict hundreds of millions of structures from metagenomics. Thus, every analysis that is done now on sets of sequences, such as homology-based inference of annotations, phylogeny and many more, would benefit from the much richer information that lies within the three-dimensional coordinates of protein structures. Novel algorithms that we are developing, such as FoldSeek, will allow these analyses to happen efficiently at a low carbon-footprint. Lastly, algorithms like PetaSearch will enable the analysis of some of the largest data sets in biology.

Various tools in the MMseqs2 family. From left to right and top to bottom: MMseqs2, Plass, MetaEuk, ColabFold, Linclust, MMseqs2 App, SpacePHARER, FoldSeek. Art by Yuna Kwon (MMseqs2, Linclust, Server), Lee Rang (Plass), Anna Papadopoulou (MetaEuk, SpacePHARER) and Doyoon Kim (FoldSeek, ColabFold).

References

A. Aevarsson, A.-K. Kaczorowska, B. T. Adalsteinsson, J. Ahlqvist, S. Al-Karadaghi, J. Altenbuchner, H. Arsin, Ú. Á. Átlasson, D. Brandt, M. Cichowicz-Cieślak, K. A. S. Cornish, J. Courtin, S. Dabrowski, H. Dahle, S. Djeffane, S. Dorawa, J. Dusaucy, F. Enault, A.-E. Fedøy, S. Freitag-Pohl, O. H. Fridjonsson, C. Galiez, E. Glomsaker, M. Guérin, S. E. Gundesø, E. E. Gudmundsdóttir, H. Gudmundsson, M. Håkansson, C. Henke, A. Helleux, J. R. Henriksen, S. Hjörleifdóttir, G. O. Hreggvidsson, A. Jasilionis, A. Jochheim, I. Jónsdóttir, L. B. Jónsdóttir, A. Jurczak-Kurek, T. Kaczorowski, J. Kalinowski, L. P. Kozlowski, M. Krupovic, K. Kwiatkowska-Semrau, O. Lanes, J. Lange, J. Lebrat, J. Linares-Pastén, Y. Liu, S. A. Lorentsen, T. Lutterman, T. Mas, W. Merré, M. Mirdita, A. Morzywołek, E. O. Ndela, E. N. Karlsson, E. Olgudóttir, C. Pedersen, F. Perler, S. K. Pétursdóttir, M. Plotka, E. Pohl, D. Prangishvili, J. L. Ray, B. Reynisson, T. Róbertsdóttir, R.-A. Sandaa, A. Sczyrba, S. Skírnisdóttir, J. Söding, T. Solstad, I. H. Steen, S. K. Stefánsson, M. Steinegger, K. S. Overå, B. Striberny, A. Svensson, M. Szadkowska, E. J. Tarrant, P. Terzian, M. Tourigny, T. van den Bergh, J. Vanhalst, J. Vincent, B. Vroling, B. Walse, L. Wang, H. Watzlawick, M. Welin, O. Werbowy, E. Wons, and R. Zhang. Going to extremes - a metagenomic journey into the dark matter of life. *FEMS Microbiol. Lett.*, 368(12):fnab067, 2021.

A. Almeida, S. Nayfach, M. Boland, F. Strozzi, M. Beracochea, Z. J. Shi, K. S. Pollard, E. Sakharova, D. H. Parks, P. Hugenholtz, N. Segata, N. C. Kyrpides, and R. D. Finn. A unified catalog of 204,938 reference genomes from the human gut microbiome. *Nat. Biotechnol.*, 39(1):105–114, 2021.

M. AlQuraishi. Machine learning in protein structure prediction. *Curr. Opin. Chem. Biol.*, 65:1–8, 2021a.

M. AlQuraishi. The AlphaFold2 Method Paper: A Fount of Good Ideas, 2021b. URL `https://moalquraishi.wordpress.com/2021/07/25/the-alphafold2-method-paper-a-fount-of-good-ideas`. [Accessed: 2021-11-23].

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, 1990.

S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25 (17):3389–3402, 1997.

C. B. Anfinsen. Principles that Govern the Folding of Protein Chains. *Science*, 181(4096):223–230, 1973.

M. Anson. Protein Denaturation and the Properties of Protein Groups. In *Adv. Protein Chem.*, pages 361–386. Elsevier, 1945. doi: 10.1016/S0065-3233(08)60629-4.

A. Bairoch and B. Boeckmann. The SWISS-PROT protein sequence data bank. *Nucleic Acids Res.*, 19 (suppl):2247–2249, 1991.

A. Bateman, M.-J. Martin, S. Orchard, M. Magrane, R. Agivetova, S. Ahmad, E. Alpi, E. H. Bowler-Barnett, R. Britto, B. Bursteinas, H. Bye-A-Jee, R. Coetzee, A. Cukura, A. Da Silva, P. Denny, T. Dogan, T. Ebenezer, J. Fan, L. G. Castro, P. Garmiri, G. Georghiou, L. Gonzales, E. Hatton-Ellis, A. Hussein, A. Ignatchenko, G. Insana, R. Ishtiaq, P. Jokinen, V. Joshi, D. Jyothi, A. Lock, R. Lopez, A. Luciani, J. Luo, Y. Lussi, A. MacDougall, F. Madeira, M. Mahmoudy, M. Menchi, A. Mishra, K. Moulang, A. Nightingale,

C. S. Oliveira, S. Pundir, G. Qi, S. Raj, D. Rice, M. R. Lopez, R. Saidi, J. Sampson, T. Sawford, E. Speretta, E. Turner, N. Tyagi, P. Vasudev, V. Volynkin, K. Warner, X. Watkins, R. Zaru, H. Zellner, A. Bridge, S. Poux, N. Redaschi, L. Aimo, G. Argoud-Puy, A. Auchincloss, K. Axelsen, P. Bansal, D. Baratin, M.-C. Blatter, J. Bolleman, E. Boutet, L. Breuza, C. Casals-Casas, E. de Castro, K. C. Echioukh, E. Coudert, B. Cuche, M. Doche, D. Dornevil, A. Estreicher, M. L. Famiglietti, M. Feuermann, E. Gasteiger, S. Gehant, V. Gerritsen, A. Gos, N. Gruaz-Gumowski, U. Hinz, C. Hulo, N. Hyka-Nouspikel, F. Jungo, G. Keller, A. Kerhornou, V. Lara, P. Le Mercier, D. Lieberherr, T. Lombardot, X. Martin, P. Masson, A. Morgat, T. B. Neto, S. Paesano, I. Pedruzzi, S. Pilbout, L. Pourcel, M. Pozzato, M. Pruess, C. Rivoire, C. Sigrist, K. Sonesson, A. Stutz, S. Sundaram, M. Tognolli, L. Verbregue, C. H. Wu, C. N. Arighi, L. Arminski, C. Chen, Y. Chen, J. S. Garavelli, H. Huang, K. Laiho, P. McGarvey, D. A. Natale, K. Ross, C. R. Vinayaka, Q. Wang, Y. Wang, L.-S. Yeh, J. Zhang, P. Ruch, and D. Teodoro. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.*, 49(D1):D480–D489, 2021.

H. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide Protein Data Bank. *Nat. Struct. Mol. Biol.*, 10(12):980–980, 2003.

M. Bernhofer, C. Dallago, T. Karl, V. Satagopam, M. Heinzinger, M. Littmann, T. Olenyi, J. Qiu, K. Schütze, G. Yachdav, H. Ashkenazy, N. Ben-Tal, Y. Bromberg, T. Goldberg, L. Kajan, S. O'Donoghue, C. Sander, A. Schafferhans, A. Schlessinger, G. Vriend, M. Mirdita, P. Gawron, W. Gu, Y. Jarosz, C. Trefois, M. Steinegger, R. Schneider, and B. Rost. PredictProtein - Predicting Protein Structure and Function for 29 Years. *Nucleic Acids Res.*, 49(W1):W535–W540, 2021.

R. Bonneau and D. Baker. Ab Initio Protein Structure Prediction: Progress and Prospects. *Annu. Rev. Biophys. Biomol. Struct.*, 30(1):173–189, 2001.

F. P. Breitwieser and S. L. Salzberg. Pavian: interactive analysis of metagenomics data for microbiome studies and pathogen identification. *Bioinformatics*, 36(4):1303–1304, 2020.

H. Brinkerhoff, A. S. W. Kang, J. Liu, A. Aksimentiev, and C. Dekker. Infinite re-reading of single proteins at single-amino-acid resolution using nanopore sequencing. *bioRxiv*, page 2021.07.13.452225, 2021.

B. Buchfink, C. Xie, and D. H. Huson. Fast and sensitive protein alignment using DIAMOND. *Nat. Methods*, 12(1):59–60, 2015.

B. Buchfink, K. Reuter, and H.-G. Drost. Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nat. Methods*, 18(4):366–368, 2021.

C. Burks, J. W. Fickett, W. B. Goad, M. Kanehisa, F. I. Lewitter, W. P. Rindone, C. D. Swindell, C.-S. Tung, and H. S. Bilofsky. The GenBank nucleic acid sequence database. *Bioinformatics*, 1(4):225–233, 1985.

C. H. Calisher. Taxonomy: what's in a name? Doesn't a rose by any other name smell as sweet? *Croat. Med. J.*, 48(2):268–270, 2007.

E. Callaway. DeepMind's AI predicts structures for a vast trove of proteins. *Nature*, 595(7869):635–635, 2021.

Q. Carradec, E. Pelletier, C. Da Silva, A. Alberti, Y. Seeleuthner, R. Blanc-Mathieu, G. Lima-Mendez, F. Rocha, L. Tirichine, K. Labadie, A. Kirilovsky, A. Bertrand, S. Engelen, M.-A. Madoui, R. Méheust, J. Poulain, S. Romac, D. J. Richter, G. Yoshikawa, C. Dimier, S. Kandels-Lewis, M. Picheral, S. Searson, Tara Oceans Coordinators, O. Jaillon, J.-M. Aury, E. Karsenti, M. B. Sullivan, S. Sunagawa, P. Bork, F. Not, P. Hingamp, J. Raes, L. Guidi, H. Ogata, C. de Vargas, D. Iudicone, C. Bowler, and P. Wincker. A global ocean atlas of eukaryotic genes. *Nat. Commun.*, 9(1):373, 2018.

M. Cobb. 60 years ago, Francis Crick changed the logic of biology. *PLoS Biol.*, 15(9):e2003243, 2017.

G. Cornilescu, J. L. Marquardt, M. Ottiger, and A. Bax. Validation of Protein Structure from Anisotropic Carbonyl Chemical Shifts in a Dilute Liquid Crystalline Phase. *J. Am. Chem. Soc.*, 120(27):6836–6837, 1998.

G. Cornilescu, J. Marquardt, M. Ottiger, and A. Bax. UBIQUITIN NMR STRUCTURE. 1999. doi: 10.2210/pdb1d3z/pdb.

M. Dayhoff, R. Schwartz, and B. Orcutt. A model of Evolutionary Change in Proteins. In *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Washington, D.C., 5 edition, 1978. ISBN 9780912466071.

N. K. Devanga Ragupathi, D. P. Muthuirulandi Sethuvel, F. Y. Inbanathan, and B. Veeraraghavan. Accurate differentiation of Escherichia coli and Shigella serogroups: challenges and strategies. *New Microbes New Infect.*, 21:58–62, 2018.

A. K. Dunker, M. M. Babu, E. Barbar, M. Blackledge, S. E. Bondos, Z. Dosztányi, H. J. Dyson, J. Forman-Kay, M. Fuxreiter, J. Gsponer, K.-H. Han, D. T. Jones, S. Longhi, S. J. Metallo, K. Nishikawa, R. Nussinov, Z. Obradovic, R. V. Pappu, B. Rost, P. Selenko, V. Subramaniam, J. L. Sussman, P. Tompa, and V. N. Uversky. What's in a name? Why these proteins are intrinsically disordered. *Intrinsically Disord. Proteins*, 1(1):e24157, 2013.

S. R. Eddy. What is dynamic programming? *Nat. Biotechnol.*, 22(7):909–910, 2004.

S. R. Eddy. Accelerated Profile HMM Searches. *PLoS Comput. Biol.*, 7(10):e1002195, 2011.

R. C. Edgar, J. Taylor, T. Altman, P. Barbera, D. Meleshko, V. Lin, D. Lohr, G. Novakovsky, B. Al-Shayeb, J. F. Banfield, A. Korobeynikov, R. Chikhi, and A. Babaian. Petabase-scale sequence alignment catalyses viral discovery. *bioRxiv*, page 2020.08.07.241729, 2020.

P. Edman, E. Högfeldt, L. G. Sillén, and P.-O. Kinell. Method for Determination of the Amino Acid Sequence in Peptides. *Acta Chem. Scand.*, 4:283–293, 1950.

I. Elias. Settling the Intractability of Multiple Alignment. *J. Comput. Biol.*, 13(7):1323–1339, 2006.

L. Eme, S. C. Sharpe, M. W. Brown, and A. J. Roger. On the Age of Eukaryotes: Evaluating Evidence from Fossils and Molecular Clocks. *Cold Spring Harb. Perspect. Biol.*, 6(8):a016139–a016139, 2014.

R. Evans, M. O'Neill, A. Pritzel, N. Antropova, A. W. Senior, T. Green, A. Žídek, R. Bates, S. Blackwell, J. Yim, O. Ronneberger, S. Bodenstein, M. Zielinski, A. Bridgland, A. Potapenko, A. Cowie, K. Tunyasuvunakool, R. Jain, E. Clancy, P. Kohli, J. Jumper, and D. Hassabis. Protein complex prediction with AlphaFold-Multimer. *bioRxiv*, page 2021.10.04.463034, 2021.

M. Farrar. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23(2):156–161, 2007.

S. Federhen. The NCBI Taxonomy database. *Nucleic Acids Res.*, 40(D1):D136–D143, 2012.

D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *J. Mol. Evol.*, 25(4):351–360, 1987.

D. Floyer. QLC Flash HAMRs HDD, 2021. URL `https://wikibon.com/qlc-flash-hamrs-hdd`. [Accessed: 2021-12-20].

R. O. Fox, P. A. Evans, and C. M. Dobson. Multiple conformations of a protein demonstrated by magnetization transfer NMR spectroscopy. *Nature*, 320(6058):192–194, 1986.

A. C. D. Fuchs, L. Maldoner, M. Wojtynek, M. D. Hartmann, and J. Martin. Rpn11-mediated ubiquitin processing in an ancestral archaeal ubiquitination system. *Nat. Commun.*, 9(1):2696, 2018.

F. Gabler, S. Nam, S. Till, M. Mirdita, M. Steinegger, J. Söding, A. N. Lupas, and V. Alva. Protein Sequence Analysis Using the MPI Bioinformatics Toolkit. *Curr. Protoc. Bioinformatics*, 72(1):e108, 2020.

T. Glasmachers. Limits of End-to-End Learning. *arXiv*, page 1704.08305, 2017.

U. Göbel, C. Sander, R. Schneider, and A. Valencia. Correlated mutations and residue contacts in proteins. *Proteins*, 18(4):309–317, 1994.

J. Goris, K. T. Konstantinidis, J. A. Klappenbach, T. Coenye, P. Vandamme, and J. M. Tiedje. DNA–DNA hybridization values and their relationship to whole-genome sequence similarities. *Int. J. Syst. Evol. Microbiol.*, 57(1):81–91, 2007.

O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162(3):705–708, 1982.

B. Grüning, R. Dale, A. Sjödin, B. A. Chapman, J. Rowe, C. H. Tomkins-Tinch, R. Valieris, and J. Köster. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods*, 15(7): 475–476, 2018.

J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman. Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chem. Biol.*, 5(10):R245–R249, 1998.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv*, page 1512.03385, 2015.

S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, 89(22):10915–10919, 1992.

L. Heo and M. Feig. Experimental accuracy in protein structure refinement via molecular dynamics simulations. *Proc. Natl. Acad. Sci.*, 115(52):13276–13281, 2018a.

L. Heo and M. Feig. What makes it difficult to refine protein models further via molecular dynamics simulations? *Proteins*, 86(S1):177–188, 2018b.

R. T. Hersh, R. V. Eck, and M. O. Dayhoff. Atlas of Protein Sequence and Structure, 1966. *Syst. Biol.*, 16 (3):262–263, 1967.

J. Hey, W. M. Fitch, and F. J. Ayala. Systematics and the origin of species: An introduction. *Proc. Natl. Acad. Sci.*, 102(Supplement 1):6515–6519, 2005.

A. Hildebrand, M. Remmert, A. Biegert, and J. Söding. Fast and accurate automatic structure prediction with HHpred. *Proteins*, 77(S9):128–132, 2009.

F. Hille, H. Richter, S. P. Wong, M. Bratovič, S. Ressel, and E. Charpentier. The Biology of CRISPR-Cas: Backward and Forward. *Cell*, 172(6):1239–1259, 2018.

T. Hu, N. Chitnis, D. Monos, and A. Dinh. Next-generation sequencing technologies: An overview. *Hum. Immunol.*, 82(11):801–811, 2021.

D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster. MEGAN analysis of metagenomic data. *Genome Res.*, 17(3):377–386, 2007.

K. Illergård, D. H. Ardell, and A. Elofsson. Structure is three to ten times more conserved than sequence-A study of structural response in protein cores. *Proteins*, 77(3):499–508, 2009.

D. T. Jones, T. Singh, T. Kosciolek, and S. Tetchner. MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, 31(7):999–1006, 2015.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021a.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Applying and improving AlphaFold at CASP14. *Proteins*, 89(12):1711–1721, 2021b.

W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–637, 1983.

H. Kamisetty, S. Ovchinnikov, and D. Baker. Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. *Proc. Natl. Acad. Sci.*, 110(39):15674–15679, 2013.

S. Karthikeyan, A. Nguyen, D. McDonald, Y. Zong, N. Ronquillo, J. Ren, J. Zou, S. Farmer, G. Humphrey, D. Henderson, T. Javidi, K. Messer, C. Anderson, R. Schooley, N. K. Martin, and R. Knight. Rapid, Large-Scale Wastewater Surveillance and Automated Reporting System Enable Early Detection of Nearly 85% of COVID-19 Cases on a University Campus. *mSystems*, 6(4):e00793–21, 2021.

J. C. Kasmanas, A. Bartholomäus, F. B. Corrêa, T. Tal, N. Jehmlich, G. Herberth, M. von Bergen, P. F. Stadler, A. C. P. de Leon Ferreira de Carvalho, and U. N. da Rocha. HumanMetagenomeDB: A public repository of curated and standardized metadata for human metagenomes. *Nucleic Acids Res.*, 49(D1): D743–D750, 2021.

D. E. Kim, F. DiMaio, R. Yu-Ruei Wang, Y. Song, and D. Baker. One contact for every twelve residues allows robust and accurate topology-level protein structure modeling. *Proteins*, 82:208–218, 2014.

S. Krishna and N. V. Grishin. Structurally Analogous Proteins Do Exist! *Structure*, 12(7):1125–1127, 2004.

A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult. Critical assessment of methods of protein structure prediction (CASP)—Round XIV. *Proteins*, 89(12):1607–1617, 2021.

B. Kuhlman and P. Bradley. Advances in protein structure prediction and design. *Nat. Rev. Mol. Cell Biol.*, 20(11):681–697, 2019.

L. Lannelongue, J. Grealey, and M. Inouye. Green Algorithms: Quantifying the Carbon Footprint of Computation. *Adv. Sci.*, 8(12):2100707, 2021.

M. Levitt and A. Warshel. Computer simulation of protein folding. *Nature*, 253(5494):694–698, 1975.

E. Levy Karin, M. Mirdita, and J. Söding. MetaEuk-sensitive, high-throughput gene discovery, and annotation for large-scale eukaryotic metagenomics. *Microbiome*, 8(1):48, 2020.

D. J. Lipman and W. R. Pearson. Rapid and Sensitive Protein Similarity Searches. *Science*, 227(4693): 1435–1441, 1985.

F. Madeira, Y. M. Park, J. Lee, N. Buso, T. Gur, N. Madhusoodanan, P. Basutkar, A. R. N. Tivey, S. C. Potter, R. D. Finn, and R. Lopez. The EMBL-EBI search and sequence analysis tools APIs in 2019. *Nucleic Acids Res.*, 47(W1):W636–W641, 2019.

M. Manni, M. R. Berkeley, M. Seppey, F. A. Simão, and E. M. Zdobnov. BUSCO Update: Novel and Streamlined Workflows along with Broader and Deeper Phylogenetic Coverage for Scoring of Eukaryotic, Prokaryotic, and Viral Genomes. *Mol. Biol. Evol.*, 38(10):4647–4654, 2021.

J. D. Marth. A unified vision of the building blocks of life. *Nat. Cell Biol.*, 10(9):1015–1015, 2008.

J. C. McCallum. Memory Prices 1957+, 2021. URL `https://jcmit.net/memoryprice.htm`. [Accessed: 2021-12-06].

P. Menzel, K. L. Ng, and A. Krogh. Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nat. Commun.*, 7(1):11257, 2016.

F. Meyer, A. Fritz, Z.-L. Deng, D. Koslicki, A. Gurevich, G. Robertson, M. Alser, D. Antipov, F. Beghini, D. Bertrand, J. J. Brito, C. Brown, J. Buchmann, A. Buluç, B. Chen, R. Chikhi, P. T. Clausen, A. Cristian, P. W. Dabrowski, A. E. Darling, R. Egan, E. Eskin, E. Georganas, E. Goltsman, M. A. Gray, L. H. Hansen, S. Hofmeyr, P. Huang, L. Irber, H. Jia, T. S. Jørgensen, S. D. Kieser, T. Klemetsen, A. Kola, M. Kolmogorov, A. Korobeynikov, J. Kwan, N. LaPierre, C. Lemaitre, C. Li, A. Limasset, F. Malcher-Miranda, S. Mangul, V. R. Marcelino, C. Marchet, P. Marijon, D. Meleshko, D. R. Mende, A. Milanese, N. Nagarajan, J. Nissen, S. Nurk, L. Oliker, L. Paoli, P. Peterlongo, V. C. Piro, J. S. Porter, S. Rasmussen, E. R. Rees, K. Reinert, B. Renard, E. M. Robertsen, G. L. Rosen, H.-J. Ruscheweyh, V. Sarwal, N. Segata, E. Seiler, L. Shi, F. Sun, S. Sunagawa, S. J. Sørensen, A. Thomas, C. Tong, M. Trajkovski, J. Tremblay, G. Uritskiy, R. Vicedomini, Z. Wang, Z. Wang, Z. Wang, A. Warren, N. P. Willassen, K. Yelick, R. You, G. Zeller, Z. Zhao, S. Zhu, J. Zhu, R. Garrido-Oter, P. Gastmeier, S. Hacquard, S. Häußler, A. Khaledi, F. Maechler, F. Mesny, S. Radutoiu, P. Schulze-Lefert, N. Smit, T. Strowig, A. Bremges, A. Sczyrba, and A. C. McHardy. Critical Assessment of Metagenome Interpretation - the second round of challenges. *bioRxiv*, page 2021.07.12.451567, 2021.

M. Mirdita, L. von den Driesch, C. Galiez, M. J. Martin, J. Söding, and M. Steinegger. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Res.*, 45(D1):D170–D176, 2017.

M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov, and M. Steinegger. ColabFold - Making protein folding accessible to all. *bioRxiv*, page 2021.08.15.456425, 2021.

J. Mistry, S. Chuguransky, L. Williams, M. Qureshi, G. A. Salazar, E. L. L. Sonnhammer, S. C. E. Tosatto, L. Paladin, S. Raj, L. J. Richardson, R. D. Finn, and A. Bateman. Pfam: The protein families database in 2021. *Nucleic Acids Res.*, 49(D1):D412–D419, 2021.

A. L. Mitchell, A. Almeida, M. Beracochea, M. Boland, J. Burgin, G. Cochrane, M. R. Crusoe, V. Kale, S. C. Potter, L. J. Richardson, E. Sakharova, M. Scheremetjew, A. Korobeynikov, A. Shlemov, O. Kunyavskaya, A. Lapidus, and R. D. Finn. MGnify: The microbiome analysis resource in 2020. *Nucleic Acids Res.*, 48 (D1):D570–D578, 2020.

J. Moult, J. T. Pedersen, R. Judson, and K. Fidelis. A large-scale experiment to assess protein structure prediction methods. *Proteins*, 23(3):ii–iv, 1995.

A. E. Murray, J. Freudenstein, S. Gribaldo, R. Hatzenpichler, P. Hugenholtz, P. Kämpfer, K. T. Konstantinidis, C. E. Lane, R. T. Papke, D. H. Parks, R. Rossello-Mora, M. B. Stott, I. C. Sutcliffe, J. C. Thrash, S. N. Venter, W. B. Whitman, S. G. Acinas, R. I. Amann, K. Anantharaman, J. Armengaud, B. J. Baker, R. A. Barco, H. B. Bode, E. S. Boyd, C. L. Brady, P. Carini, P. S. G. Chain, D. R. Colman, K. M. DeAngelis, M. A. de los Rios, P. Estrada-de los Santos, C. A. Dunlap, J. A. Eisen, D. Emerson, T. J. G. Ettema, D. Eveillard, P. R. Girguis, U. Hentschel, J. T. Hollibaugh, L. A. Hug, W. P. Inskeep, E. P. Ivanova, H.-P. Klenk, W.-J. Li, K. G. Lloyd, F. E. Löffler, T. P. Makhalanyane, D. P. Moser, T. Nunoura, M. Palmer, V. Parro, C. Pedrós-Alió, A. J. Probst, T. H. M. Smits, A. D. Steen, E. T. Steenkamp, A. Spang, F. J. Stewart, J. M. Tiedje, P. Vandamme, M. Wagner, F.-P. Wang, P. Yarza, B. P. Hedlund, and A.-L. Reysenbach. Roadmap for naming uncultivated Archaea and Bacteria. *Nat. Microbiol.*, 5(8):987–994, 2020.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, 1970.

B. D. Ondov, N. H. Bergman, and A. M. Phillippy. Interactive metagenomic visualization in a Web browser. *BMC Bioinform.*, 12(1):385, 2011.

S. Pan, C. Zhu, X.-M. Zhao, and L. P. Coelho. SemiBin: Incorporating information from reference genomes with semi-supervised deep learning leads to better metagenomic assembled genomes (MAGs). *bioRxiv*, page 2021.08.16.456517, 2021.

D. H. Parks, M. Chuvochina, D. W. Waite, C. Rinke, A. Skarshewski, P. A. Chaumeil, and P. Hugenholtz. A standardized bacterial taxonomy based on genome phylogeny substantially revises the tree of life. *Nat. Biotechnol.*, 36(10):996, 2018.

D. H. Parks, M. Chuvochina, C. Rinke, A. J. Mussig, P.-A. Chaumeil, and P. Hugenholtz. GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. *Nucleic Acids Res.*, page gkab776, 2021.

A. Payne, N. Holmes, V. Rakyan, and M. Loose. BulkVis: a graphical viewer for Oxford nanopore bulk FAST5 files. *Bioinformatics*, 35(13):2193–2198, 2019.

D. Rajasekaran, S. Zierow, M. Syed, R. Bucala, V. Bhandari, and E. J. Lolis. Targeting distinct tautomerase sites of D-DT and MIF with a single molecule for inhibition of neutrophil lung recruitment. *FASEB J.*, 28 (11):4961–71, 2014.

G. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *J. Mol. Biol.*, 7(1):95–99, 1963.

J. S. Richardson. Early ribbon drawings of proteins. *Nat. Struct. Biol.*, 7(8):624–625, 2000.

S. L. Robinson, J. Piel, and S. Sunagawa. A roadmap for metagenomic enzyme discovery. *Nat. Prod. Rep.*, 38(11):1994–2023, 2021.

B. Rost. Twilight zone of protein sequence alignments. *Protein Eng. Des. Sel.*, 12(2):85–94, 1999.

R. A. Sanford, K. G. Lloyd, K. T. Konstantinidis, and F. E. Löffler. Microbial Taxonomy Run Amok. *Trends Microbiol.*, 29(5):394–404, 2021.

F. Sanger and A. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J. Mol. Biol.*, 94(3):441–448, 1975.

E. W. Sayers, J. Beck, E. E. Bolton, D. Bourexis, J. R. Brister, K. Canese, D. C. Comeau, K. Funk, S. Kim, W. Klimke, A. Marchler-Bauer, M. Landrum, S. Lathrop, Z. Lu, T. L. Madden, N. O'Leary, L. Phan, S. H. Rangwala, V. A. Schneider, Y. Skripchenko, J. Wang, J. Ye, B. W. Trawick, K. D. Pruitt, and S. T. Sherry. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, 49(D1): D10–D17, 2021a.

E. W. Sayers, M. Cavanaugh, K. Clark, K. D. Pruitt, C. L. Schoch, S. T. Sherry, and I. Karsch-Mizrachi. GenBank. *Nucleic Acids Res.*, 49(D1):D92–D96, 2021b.

J. Schaarschmidt, B. Monastyrskyy, A. Kryshtafovych, and A. M. Bonvin. Assessment of contact predictions in CASP12: Co-evolution and deep learning coming of age. *Proteins*, 86:51–66, 2018.

T. D. Schneider and R. Stephens. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res.*, 18(20):6097–6100, 1990.

A. Sczyrba, P. Hofmann, P. Belmann, D. Koslicki, S. Janssen, J. Dröge, I. Gregor, S. Majda, J. Fiedler, E. Dahms, A. Bremges, A. Fritz, R. Garrido-Oter, T. S. Jørgensen, N. Shapiro, P. D. Blood, A. Gurevich, Y. Bai, D. Turaev, M. Z. Demaere, R. Chikhi, N. Nagarajan, C. Quince, F. Meyer, M. Balvočiutė, L. H. Hansen, S. J. Sørensen, B. K. Chia, B. Denis, J. L. Froula, Z. Wang, R. Egan, D. Don Kang, J. J. Cook, C. Deltel, M. Beckstette, C. Lemaitre, P. Peterlongo, G. Rizk, D. Lavenier, Y. W. Wu, S. W. Singer, C. Jain, M. Strous, H. Klingenberg, P. Meinicke, M. D. Barton, T. Lingner, H. H. Lin, Y. C. Liao, G. G. Z. Silva, D. A. Cuevas, R. A. Edwards, S. Saha, V. C. Piro, B. Y. Renard, M. Pop, H. P. Klenk, M. Göker, N. C. Kyrpides, T. Woyke, J. A. Vorholt, P. Schulze-Lefert, E. M. Rubin, A. E. Darling, T. Rattei, and A. C. McHardy. Critical Assessment of Metagenome Interpretation - A benchmark of metagenomics software. *Nat. Methods*, 14 (11):1063–1071, 2017.

S. Seemayer, M. Gruber, and J. Söding. CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

T. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147(1): 195–197, 1981.

M. Steinegger and S. L. Salzberg. Terminating contamination: large-scale search identifies more than 2,000,000 contaminated entries in GenBank. *Genome Biol.*, 21(1):115, 2020.

M. Steinegger and J. Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, 35(11):1026–1028, 2017.

M. Steinegger, M. Meier, M. Mirdita, H. Vöhringer, S. J. Haunsberger, and J. Söding. HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinform.*, 20(1):473, 2019a.

M. Steinegger, M. Mirdita, and J. Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods*, 16(7):603–606, 2019b.

P. Stothard. The Sequence Manipulation Suite: JavaScript Programs for Analyzing and Formatting Protein and DNA Sequences. *Biotechniques*, 28(6):1102–1104, 2000.

B. J. Strasser. Collecting, Comparing, and Computing Sequences: The Making of Margaret O. Dayhoff's Atlas of Protein Sequence and Structure, 1954–1965. *J. Hist. Biol.*, 43(4):623–660, 2010.

H. Sugimoto, M. Taniguchi, A. Nakagawa, and I. Tanaka. D-DOPACHROME TAUTOMERASE. 1998. doi: 10.2210/pdb1dpt/pdb.

H. Sugimoto, M. Taniguchi, A. Nakagawa, I. Tanaka, M. Suzuki, and J. Nishihira. Crystal structure of human D-Dopachrome tautomerase, a homologue of macrophage migration inhibitory factor, at 1.54 Å resolution. *Biochemistry*, 38(11):3268–3279, 1999.

K. Tunyasuvunakool, J. Adler, Z. Wu, T. Green, M. Zielinski, A. Žídek, A. Bridgland, A. Cowie, C. Meyer, A. Laydon, S. Velankar, G. J. Kleywegt, A. Bateman, R. Evans, A. Pritzel, M. Figurnov, O. Ronneberger, R. Bates, S. A. A. Kohl, A. Potapenko, A. J. Ballard, B. Romera-Paredes, S. Nikolov, R. Jain, E. Clancy, D. Reiman, S. Petersen, A. W. Senior, K. Kavukcuoglu, E. Birney, P. Kohli, J. Jumper, and D. Hassabis. Highly accurate protein structure prediction for the human proteome. *Nature*, 596(7873):590–596, 2021.

G. W. Tyson, J. Chapman, P. Hugenholtz, E. E. Allen, R. J. Ram, P. M. Richardson, V. V. Solovyev, E. M. Rubin, D. S. Rokhsar, and J. F. Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428(6978):37–43, 2004.

M. Varadi, S. Anyango, M. Deshpande, S. Nair, C. Natassia, G. Yordanova, D. Yuan, O. Stroe, G. Wood, A. Laydon, A. Žídek, T. Green, K. Tunyasuvunakool, S. Petersen, J. Jumper, E. Clancy, R. Green, A. Vora, M. Lutfi, M. Figurnov, A. Cowie, N. Hobbs, P. Kohli, G. Kleywegt, E. Birney, D. Hassabis, and S. Velankar. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Res.*, page gkab1061, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *arXiv*, page 1706.03762, 2017.

J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. Gabor Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu-Threideh, E. Beasley, et al. The Sequence of the Human Genome. *Science*, 291(5507):1304–1351, 2001.

F. A. Von Meijenfeldt, K. Arkhipova, D. D. Cambuy, F. H. Coutinho, and B. E. Dutilh. Robust taxonomic classification of uncharted microbial sequences and bins with CAT and BAT. *Genome Biol.*, 20(1):217, 2019.

S. Wang, S. Sun, and J. Xu. Analysis of deep learning methods for blind protein contact prediction in CASP12. *Proteins*, 86(S1):67–77, 2018.

Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au. Nanopore sequencing technology, bioinformatics and applications. *Nat. Biotechnol.*, 39(11):1348–1365, 2021.

A. M. Waterhouse, J. B. Procter, D. M. A. Martin, M. Clamp, and G. J. Barton. Jalview Version 2–a multiple sequence alignment editor and analysis workbench. *Bioinformatics*, 25(9):1189–91, 2009.

K. A. Wetterstrand. The Cost of Sequencing a Human Genome, 2021a. URL `https://genome.gov/sequencingcosts`. [Accessed: 2021-11-12].

K. A. Wetterstrand. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP) , 2021b. URL `https://www.genome.gov/sequencingcostsdata`. [Accessed: 2021-11-12].

M. Wojtynek, J. Martin, and M. Hartmann. Caldiarchaeum Subterraneum Ubiquitin. 2018. doi: 10.2210/pdb6fj7/pdb.

D. E. Wood and S. L. Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, 15(3):R46, 2014.

D. E. Wood, J. Lu, and B. Langmead. Improved metagenomic analysis with Kraken 2. *Genome Biol.*, 20 (1):257, 2019.

S. H. Ye, K. J. Siddle, D. J. Park, and P. C. Sabeti. Benchmarking Metagenomics Tools for Taxonomic Classification. *Cell*, 178(4):779–794, 2019.

R. Zhang, M. Mirdita, E. Levy Karin, C. Norroy, C. Galiez, and J. Söding. SpacePHARER: sensitive identification of phages from CRISPR spacers in prokaryotic hosts. *Bioinformatics*, 37(19):3364–3366, 2021.

M. Zhao, W.-P. Lee, E. P. Garrison, and G. T. Marth. SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications. *PLoS One*, 8(12):e82138, 2013.

S. Zierow and E. Lolis. D-dopachrome tautomerase (D-DT)/macrophage migration inhibitory factor 2 (MIF2) complexed with inhibitor 4-IPP. 2009. doi: 10.2210/pdb3kan/pdb.

G. Zuo, Z. Xu, and B. Hao. Shigella Strains Are Not Clones of Escherichia coli but Sister Species in the Genus Escherichia. *Genom. Proteom. Bioinform.*, 11(1):61–65, 2013.

Appendix

A1 MMseqs2 User Guide – Table of Contents

The MMseqs2 user-guide is available online at `github.com/soedinglab/MMseqs2/wiki`. Due to its extensive size, I included only the table of contents. Writing and maintaining the MMseqs2 user guide has been a continuous collaborative effort by the whole MMseqs2 team.

1. Summary
2. System requirements
3. Installation
   - Install MMseqs2 for Linux
   - Install MMseqs2 for macOS
   - Install MMseqs2 for Windows
   - Use the Docker image
   - Set up the Bash/Zsh command completion
   - Customizing compilation through CMake
4. Getting started
   - Usage of MMseqs2 modules
   - Easy workflows
   - Downloading databases
   - Searching
   - Clustering
   - Linclust
   - Updating a clustered database
5. Overview of folders in MMseqs2
6. Overview of important MMseqs2 modules
7. Description of workflows
   - Batch sequence searching using `mmseqs search`
   - Translated sequence searching
   - Expanded cluster searches
   - Mapping very similar sequences using `mmseqs map`
   - Clustering databases using `mmseqs cluster` or `mmseqs linclust`
     - Clustering criteria
     - Cascaded clustering
     - Clustering modes
   - Linear time clustering using `mmseqs linclust`
     - Run Linclust
   - Updating a clustering database using `mmseqs clusterupdate`
   - Taxonomy assignment
     - Terminology
     - Creating a seqTaxDB
     - Filtering a seqTaxDB

    - The concept of LCA
    - Using seqTaxDB for taxonomy assignment
    - Taxonomy output and TSV
    - Taxonomic ranks
    - Taxonomy report in Kraken or Krona style
    - Taxonomy top hit report
    - Filtering taxonomy output
    - Taxonomy annotation of search/cluster results
  - Reciprocal best hit using `mmseqs rbh`

8. Description of core modules
   - Computation of prefiltering scores using `mmseqs prefilter`
     - Set sensitivity `-s` parameter
   - Local alignment of prefiltered sequence pairs using `mmseqs align`
   - Clustering sequence database using `mmseqs clust`
9. File Formats
   - MMseqs2 database format
   - Manipulating databases
   - Sequence database format
   - Prefiltering format
   - Alignment format
     - Custom alignment format with convertalis
     - Internal alignment format
   - Clustering format
     - Internal cluster format
     - Cluster TSV format
     - Cluster FASTA-like format
     - Extract representative sequence
   - Taxonomy format
     - Internal taxonomy format
     - Taxonomy report in Kraken or Krona style
     - LCA TSV
   - Profile format
     - Parameters that affect profile construction
     - Convert a result database into a profile
     - Convert an external MSA into a profile
     - Extract consensus or sequence information from a profile
     - Convert HHsuite HMMs into a profile
   - Identifier parsing
10. Optimizing sensitivity and consumption of resources
    - Prefiltering module
      - Memory consumption
        - Database splitting runtime slowdown
      - Runtime
      - Disk space
      - Important options for tuning the memory, runtime and disk space usage
    - Alignment module
      - Memory consumption

        - Runtime
        - Disk space
    - Clustering module
        - Memory consumption
        - Runtime
        - Disk space
    - Workflows

11. How to run MMseqs2 on multiple servers using MPI
12. How to run MMseqs2 on multiple servers using batch systems
13. Frequently Asked Questions
    - How to set the right alignment coverage to cluster
    - How do parameters of CD-HIT relate to MMeqs2
    - How does MMseqs2 compute the sequence identity
    - How to restart a search or clustering workflow
    - How to control the speed of the search
    - How to find the best hit the fastest way
    - How does MMseqs2 handle low complexity
    - How to redundancy filter sequences with identical length and 100% length overlap.
    - How to add sequence identities and other alignment information to a clustering result.
    - How to run external tools for each database entry
    - How to compute a multiple alignment for each cluster
    - How to manually cascade cluster
    - How to cluster using profiles
    - How to create a HHblits database
    - How to create a target profile database (from PFAM)
    - How to cluster a graph given as TSV or m8 file
    - How to search small query sets fast
    - What is the difference between the map and search workflow
    - How to build your own MMseqs2 compatible substitution matrices
    - How to create a fake prefiltering for all-vs-all alignments
    - How to compute the lowest common ancestor (LCA) of a given set of sequences
14. Workflow control parameters
    - Search workflow
    - Clustering workflow
    - Updating workflow
15. Environment variables used by MMseqs2
16. External libraries used in MMseqs2
17. License terms