Linclust: clustering protein sequences in linear time

Martin Steinegger[1,2] & Johannes Söding[1,*]

[1]Quantitative and Computational Biology group, Max-Planck Institute for Biophysical Chemistry, Am Fassberg 11, 37077 Göttingen, Germany. [2]Department for Bioinformatics and Computational Biology, Technische Universität München, 85748 Garching, Germany

Metagenomic datasets contain billions of protein sequences that could greatly enhance large-scale prediction of protein functions and structures. Linclust can cluster sequences down to 50% pairwise sequence similarity and its runtime scales linearly with the input set size, not nearly quadratically as in conventional algorithms. We cluster 1.7 billion sequences from ∼2200 metagenomic and metatranscriptomic datasets in 30 hours on 28 cores. Free software and data available at https://mmseqs.com/**.**

In metagenomics, DNA is sequenced directly from the environment, allowing us to study the vast majority of microbial diversity that cannot be cultivated [1]. During the last decade, costs and throughput of next-generation sequencing have dropped two-fold each year, twice faster than computational costs. This enormous progress has resulted in hundreds of thousands of metagenomes and tens of billions of putative gene and protein sequences [2, 3]. Thus computing and storage costs are now dominating metagenomics [4, 5, 6]. Clustering protein sequences predicted from sequencing reads or pre-assembled contigs can significantly reduce the redundancy of sequence sets and costs of downstream analysis and storage.

CD-HIT and UCLUST [7, 8] are by far the most widely used protein clustering tools. In their greedy clustering approach each of the N input sequences is compared with the N_{clus} representative sequences of already established clusters. Runtime therefore scales as $\mathrm{O}(N \times N_{\text{clus}})$, almost quadratically with the number of input sequences. Available main memory further constrains the size of sequence sets: UCLUST requires $NL \times 10\,\mathrm{B}$ and CD-HIT $NL \times 1.5\,\mathrm{B}$, where L is the average sequence length. Linclust overcomes these limitations with a linear run time and a memory footprint of only $N \times 320\,\mathrm{B}$.

Linclust proceeds in five consecutive stages (**Fig. 1**) (Online Methods): (1) It finds exact k-mer matches (default: $k = 10$) between sequences. To increase sensitivity we use a reduced alphabet of size 14 at this stage. For each sequence, Linclust selects the m (default: 20) k-mers obtaining the lowest hash values. It thereby tends to select the same subset of k-mers in every sequence. Linclust sorts all selected k-mers lexicographically, which identifies groups of sequences sharing the same k-mer. A centre sequence is picked for each group, and (2) for every sequence in the group the Hamming distance to the centre sequence is computed for the gapless extension of the k-mer match to the sequence ends. Sequences that already now fulfill the coverage and sequence identity criteria ("safe matches")

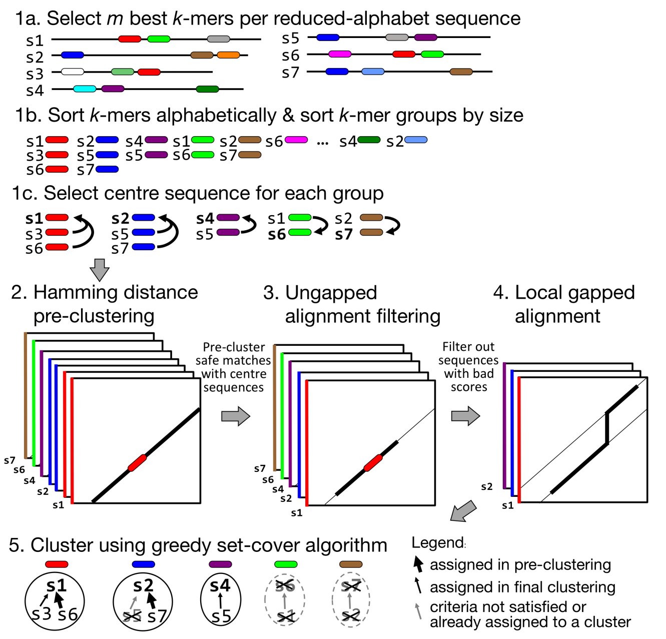
1a. Select m best k-mers per reduced-alphabet sequence
s1
s2
s3
s4
s5
s6
s7
1b. Sort k-mers alphabetically & sort k-mer groups by size
1c. Select centre sequence for each group
2. Hamming distance pre-clustering
3. Ungapped alignment filtering
4. Local gapped alignment
Pre-cluster safe matches with centre sequences
Filter out sequences with bad scores
5. Cluster using greedy set-cover algorithm
Legend
assigned in pre-clustering
assigned in final clustering
criteria not satisfied or already assigned to a cluster

FIG. 1. The five stages of Linclust. Coloured boxes represent k-mers selected as potential seeds for alignment extension in stages 2-4.

are assigned to their centre sequences. (3) For all others the maximum score for the ungapped alignment around the k-mer match is computed using the Blosum62 amino acid similarity matrix. Sequences with a sub-threshold score are filtered out. All others are aligned to their centre sequence using a vectorized implementation of local, gapped sequence alignment. Sequence pairs that satisfy the coverage and sequence identity criteria are linked. (5) Finally, the sequences are clustered using the greedy set-cover algorithm.

We measured runtimes for clustering seven protein sets. The first six were created by subsampling 1/32, 1/16, 1/8, 1/4, half and all of the 61 522 444 sequences of the UniProt database (release 2016_03). The seventh set contains the UniProt plus all sequences in reverse.

Each tool clustered these sets using a minimum pairwise sequence identity of 90%, 70% and 50%. The sequence identity in UCLUST and CD-HIT is defined as fraction of identical residues relative to the length of the shorter sequence. In Linclust, we use a highly correlated measure (Fig. S2 in [9]) that is better suited to distinguish homologous from non-homologous sequences: the local alignment score divided by the maximum length of the two aligned sequence segments. To ensure comparable acceptance criteria, we demanded in addition a minimum coverage of 90 % of the shorter by the longer sequence. We measured runtimes with the Linux time command. Benchmarks were done on a server with two Intel Xeon E5-2640v3 8-core CPUs and 128 GB RAM.

Figure 2A shows the runtimes versus sequence set size. At 50% identity, Linclust clusters the 123 million sequences 200 times faster than UCLUST and, by extrapolation,

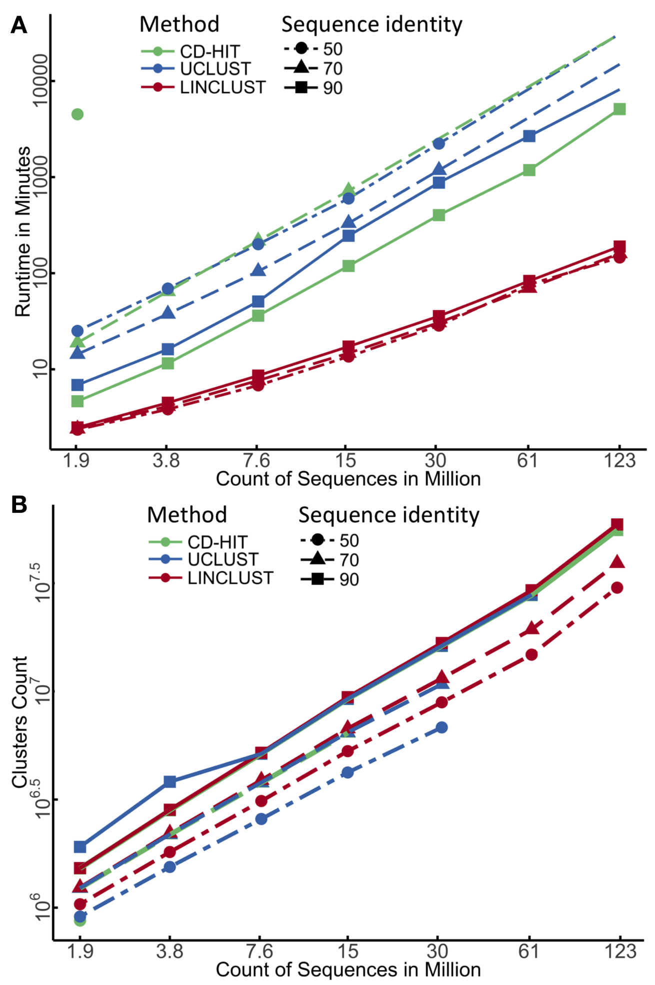

FIG. 2. (**A**) Double-logarithmic plot of runtimes versus sequence set size. Tools were run with sequence identity thresholds of 90%, 70% and 50%. Runtimes at 70 % between smallest and largest set scale with set size N as $N^{1.75}$ for CD-HIT, $N^{1.60}$ for UCLUST and $N^{1.01}$ for Linclust. (**B**) Number of clusters obtained at 90%, 70% and 50% sequence identity. Lower cluster numbers imply higher sensitivities to detect similar sequences. Some clustering runs were terminated after 48 h to save time.

more than four orders of magnitude faster than CD-HIT. At 90% identity, Linclust still clusters these sequences 43 times faster than UCLUST and 27 times faster than CD-HIT. The speedup factors grow roughly as $N^{0.6}$ and $N^{0.7}$ with the dataset size N. Stage 2 (Hamming distance) and 3 (ungapped alignment) additionally reduce the runtime by a factor 2.0 and 1.37, respectively (**Supplemental Fig. S1**)

To assess the quality of the clusterings it is sufficient to measure the number of clusters produced, because the tools' acceptance criteria for linking sequences are very similar (**Supplemental Fig. S2**). All three tools produce similar numbers of clusters at 90% and 70% sequence identity (**Fig. 2B**). Importantly, despite Linclust's linear scaling of the runtime with the number of sequences, the comparison shows that it does not suffer any loss of

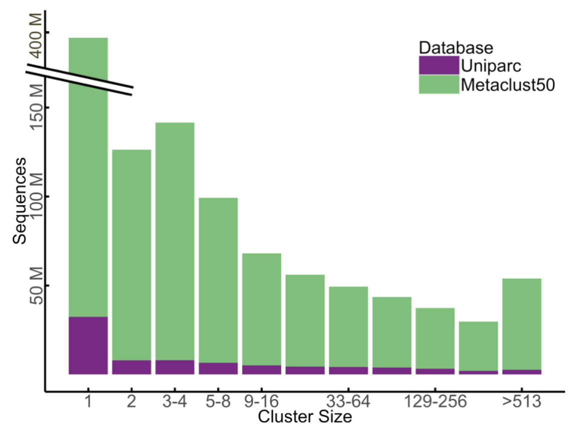

FIG. 3. Number of sequences contained in clusters with various size ranges obtained by Linclust with 50% sequence identity and a minimum coverage of 80%. Sequences in Metaclust50 clusters are shown in green, sequences from UniParc are violett.

sensitivity for growing dataset sizes.

At 50%, Linclust's sensitivity reaches its limits, producing 23% more clusters than UCLUST. But by increasing the number of k-mers selected per sequence from 20 to 80 it achieves nearly the same sensitivity at a moderate loss of speed of a factor 1.6 (**Supplemental Fig. S3**).

Linclust should enable considerable savings of computer resources in current clustering applications. Most importantly, however, it will make previously infeasible tasks possible. To demonstrate this, we applied it to cluster 1.6 billion protein sequences of metagenomic and metatranscriptomic origin [2, 10, 11] together with 123 millions sequences from the UniParc database [12].

Because many metagenomic sequences are fragmentary, we used the Hamming pre-clustering stage 2 to map fragments to longer sequences that cover at least 99% of them with a sequence identity of 95%, which yielded the redundancy-filtered set "Metaclust95". We then ran stages 3 to 5 to filter down the Metaclust95 set to 50% sequence identity using a minimum coverage of 80% of the shorter *and* longer sequence. The representative sequences from stage 5 (clustering) make up the "Metaclust50" set.

Runtimes were measured on a server with two Intel Xeon E5-2680v4 14-core CPUs and 768 GB RAM. Linclust took 17 hours to filter the 1.72 billion sequences for redundancy at 95% and remove fragments (stages 1, 2 and 5), producing 1.1 billion sequences. Clustering this Metaclust95 sequence set to 50% sequence identity using stages 3 to 5 took another 13 hours and produced 529 million clusters.

Figure 3 shows the cluster size distribution for the Metaclust50 set. We also clustered the UniParc database in exactly the same way, using the 95% redundancy and fragment filtering and then the clustering to 50%. The comparison shows that Metaclust50 has a very similar relative distribution over the cluster sizes, indicating that only a small fraction of out-of-frame or chimeric sequences are contained in Metaclust95. However, because Metaclust95 contains 14 times more sequences than UniParc95 (1.1 B versus 80 M), it has 14 times more sequences that are members of large, diverse clusters. E.g. Metaclust50 has 324 M sequences in clusters with more than 10 sequences, while UniParc has only 24 M.

Metaclust95 and Metaclust50 are, as far as we know, the largest freely available redundancy-filtered protein sequence sets. They could become a valuable resource for applications whose performance grows with the evolutionary diversity of multiple sequence alignments of protein families. (1) They will help to improve the sensitivity of profile sequence searches and to increase the fraction of annotatable sequences in genomic and metagenomic datasets [6, 10]. (2) They will boost the number protein families for which reliable structures can be predicted de novo, as shown by Ovchinnikov et al. [13], who used an as yet unpublished dataset of 2 billion metagenomic sequences. (3) They will allow us to build more accurate models to predict the effects of mutations on proteins [14].

To achieve linear runtime overall, three insights were critical. First, the identification of exact k-mer matches by sorting them, which is similar to double indexing [15], has quasi linear time scaling $\mathrm{O}(N \log N)$. Second, we showed that a small number m of well-selected k-mers per sequence is sufficient to cluster sequences down to 50% sequence identity. Third, by aligning each sequence within a group of sequences sharing a k-mer to a single "centre" sequence, we limit the number of pairwise alignments to the total number Nm of extracted k-mers. Fourth, our greedy approach of assigning centre sequences ensures that they become highly connected hub nodes around which large clusters can be built in the greedy set cover clustering stage.

We have integrated Linclust into the software package MMseqs2 (Many-to-many sequence searches) currently under development in our lab [16]. We hope Linclust and MMseqs2 will prove helpful to many researchers seeking to exploit the tremendous value of the many publicly available metagenomic and metatranscriptomic datasets.

Acknowledgements

We thank all who contributed metagenomic datasets used to build Metaclust. Some data were produced by the US Department of Energy Joint Genome Institute http://www.jgi.doe.gov/ in collaboration with the user community. This work was supported by the EU's Horizon 2020 Framework Programme (Virus-X, grant 685778) and by the Bundesministerium für Bildung und Forschung (grants e:AtheroSysMed 01ZX1313D, SysCore 0316176A). We thank M. Mirdita and C. Galiez for feedback to the manuscript.

Author contributions

MS performed research and programming, MS and JS jointly designed the research and wrote the manuscript.

[1] Rappe, M. S. & Giovannoni, S. J. The uncultured microbial majority. *Ann. Rev. Microbiol.* **57**, 369–394 (2003).

[2] Markowitz, V. M. *et al.* IMG/M 4 version of the integrated metagenome comparative analysis system. *Nucleic Acids Res.* **42**, D568–D573 (2014).

[3] Wilke, A. *et al.* The MG-RAST metagenomics database and portal in 2015. *Nucleic Acids Res.* **44**, D590–D594 (2016).

[4] Scholz, M. B., Lo, C.-C. & Chain, P. S. Next generation sequencing and bioinformatic bottlenecks: the current state of metagenomic data analysis. *Curr. Opin. Biotechnol.* **23**, 9–15 (2012).

[5] Desai, N., Antonopoulos, D., Gilbert, J. A., Glass, E. M. & Meyer, F. From genomics to metagenomics. *Curr. Opin. Biotechnol.* **23**, 72–76 (2012).

[6] Prakash, T. & Taylor, T. D. Functional assignment of metagenomic data: challenges and applications. *Brief. Bioinformatics* **13**, 711–727 (2012).

[7] Edgar, R. C. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* **26**, 2460–2461 (2010).

[8] Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28**, 3150–3152 (2012).

[9] Hauser, M., Mayer, C. & Soding, J. kClust: fast and sensitive clustering of large protein sequence databases. *BMC Bioinformatics* **14**, 248+ (2013).

[10] Sunagawa, S. *et al.* Structure and function of the global ocean microbiome. *Science* **348**, 1261359–1–9 (2015).

[11] Kodama, Y., Shumway, M. & Leinonen, R. The sequence read archive: explosive growth of sequencing data. *Nucleic Acids Res.* **40**, D54–D56 (2012).

[12] Bairoch, A. *et al.* The Universal Protein Resource (UniProt). *Nucleic Acids Res.* **33**, D154–D159 (2005).

[13] Ovchinnikov, S. *et al.* Protein structure determination using metagenome sequence data. *Science* **355**, 294–298 (2017).

[14] Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nature Biotechnology* (2017).

[15] Hauswedell, H., Singer, J. & Reinert, K. Lambda: the local aligner for massive biological data. *Bioinformatics* **30**, i349–i355 (2014).

[16] Steinegger, M. & Soeding, J. Sensitive protein sequence searching for the analysis of massive data sets. *bioRxiv* doi: 10.1101/079681 (2016).

[17] Zhao, M., Lee, W.-P., Garrison, E. P. & Marth, G. T. SSW Library: An SIMD Smith-Waterman C/C++ Library for Use in Genomic Applications. *PLoS One* **8** (2013).

[18] Hauser, M., Steinegger, M. & Söding, J. MMseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics* **32**, 1323–1330 (2016).

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

[20] Mirdita, M. *et al.* Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Res.* doi: 10.1093/nar/gkw1081 (2016).

Online methods

Linclust is composed of five stages (**Fig. 1**):

1. Finding exact k-mer matches. The first stage finds exact k-mers matches between sequences that are extended in stages 2-4. We transform the sequences into a reduced alphabet of size A (see below) to increase the number of k-mer matches and hence the k-mer sensitivity at a moderate reduction in selectivity. For each sequence we extract m k-mers as described below (Selection of k-mers). The default values are $A = 14$, $k = 10$, $m = 20$. A higher m increases sensitivity (**Supplemental Fig. S3**).

We store each extracted k-mer, its position j in the sequence, and the sequence identifier, in an array of length Nm. We sort the array by the k-mers using insertion sort from the OpenMP template library (http://freecode.com/projects/omptl). The sorted array groups sequences together that contain the same k-mer. We add group size to each array entry and resort the array by group size and, with lower priority, by k-mer. For each group, from largest to smallest, we select as the group centre the longest sequence that has not already been chosen as a centre of some previously processed group. We store for each array entry the diagonal $i - j$ of the k-mer match, where i is the positions of the k-mer in the centre sequence. Linclust requires memory equal to the size of the array, $Nm(k + 6)$ B.

2. Hamming distance pre-clustering. For each group we compute the Hamming distance (i.e., the number of mismatches) between the centre sequence and all other member sequences along the stored diagonals $i - j$ in the full amino acid alphabet. This operation is fast as it needs no random memory or cache access and uses AVX2/SSE4.1 instructions. Members that already satisfy the specified sequence identity and coverage thresholds on the entire diagonal are clustered using the greedy set-cover algorithm, removed from the set passed to stage 3, and are added to the cluster of their centre sequence after step 5.

3. Ungapped alignment filtering. For each group we compute the optimal ungapped, local alignments between the centre sequence and the member sequences along the stored diagonal $i - j$, using one-dimensional dynamic programming with the Blosum62 matrix. We filter out matches between centre and member sequences if the ungapped alignment score divided by the length of the diagonal is very low. We set a conservative threshold, such that the false negative rate is 1 %, i.e., only 1 % of the alignments below this threshold would satisfy the two criteria, sequence identity and coverage. For each combination on a grid $\{50, 55, 60, \ldots, 100\} \otimes \{0, 10, 20, \ldots, 100\}$, we determined these thresholds empirically on 4 M local alignments sampled from an all-against-all comparison of the UniProt database [12].

4. Local gapped sequence alignment. Member sequences that pass the ungapped alignment filter are aligned to their centre sequence using the AVX2/SSE4.1-vectorized alignment module with amino acid compositional bias correction from MMseqs2 [16], which builds on code from the SSW library [17]. Sequences satisfying the sequence identity and coverage thresholds are linked.

5. Clustering using greedy set cover. The sequences are clustered using the greedy set cover algorithm in MMseqs [18]. It processes the sequences in order of decreasing number of linked sequences. Each such sequence and its linked neighbours are assigned to a new cluster and removed from all lists of neighbours. The next sequence having the most linked neighbours is picked until no sequence remains unassigned. Greedy set cover is fast and yields good clusterings in practice. Its runtime is proportional to the total number of edges, which is bounded by Nm.

Reduced amino acid alphabet We iteratively constructed reduced alphabets starting from the full amino acid alphabet. At each step we merged the two letters $\{a, b\} \rightarrow a' = (a \text{ or } b)$ that conserve the maximum mutual information, $\mathrm{MI} = \sum_{x,y=1}^{A} p(x, y) \log_2 (p(a, y)/p(x)/p(y))$. Here A is the new alphabet size, $p(x)$ is the probability of observing letter x at any given position, and $p(x, y)$ is the probabilities of observing x and y aligned to each other. These probabilities are extracted from the Blosum62 matrix. When a and b are merged into a', for example, $p(a') = p(a) + p(b)$ and $p(a', y) = p(a, y) + p(b, y)$. The default alphabet with $A = 14$ merges (L,M), (I,V), (K,R), (E, Q), (A,S,T), and (F,Y).

Selection of k-mers. To be able to cluster together sequences with, e.g. 50% sequence identity, we need to find a k-mer in the reduced alphabet common to both. Because we extract only a small fraction of k-mers from each sequence, we need to avoid picking different k-mers in each sequence. Our first criterion for k-mer selection is therefore to extract the same subset of k-mers in all sequences. Second, we need to avoid positional clustering of selected k-mers in order to be sensitive to detect local homologies in every region of a sequence. Third, we would like to extract k-mers that tend to be conserved between homologous sequences. We note that the k-mers to be selected cannot simply be stored due to their sheer number ($\approx A^k\, m/L$).

We can satisfy the first two criteria by computing hash values for all k-mers in a sequence and selecting the m k-mers that obtain the lowest hash values. Since appropriate hash functions can produce values that are not correlated in any simple way with the hash keys, i.e. our k-mers, this method should randomly select k-mers from the sequences such that the same k-mers always tend to get selected in all sequences. We developed a simple 16 bit rolling hash function with good mixing properties, which we can compute very efficiently using the hash value of the previous k-mer (**Supplemental Fig. 4**).

In view of the third criterion, we experimented with combining the hash value with a k-mer conservation score $S_{\text{cons}}(x_{1:k}) = \sum_{i=1}^{k} S(x_i, x_i)/k$. This score ranks k-mers $x_{1:k}$ by the conservation of their amino acids, according to the diagonal elements of the Blosum62 substitution matrix $S(\cdot,\cdot)$. We scaled the hash function with a rectified version of the conservation score: hash-value$(x_{1:k})/\max\{1, S_{\text{cons}}(x_{1:k}) - S_{\text{offset}}\}$. Despite its intuitive appeal, we did not succeed in obtaining significant improvements and reverted to the simple hash function.

Parallelization and supported platforms. We used OpenMP to parallelize all stages except the IO-limited stage 1c (Figure 1A), by applying the "#pragma omp parallel for" directive to the loops over the input sequences (stage 1a,b) or centre sequences (stages 2, 3). Linclust supports Linux and Mac OS X and CPUs with AVX2 or SSE4.1 instructions.

Tools and command line options for benchmark comparison. We tested CD-HIT (version 4.6) with the parameters -T 16 -M 0 and -n 5 -c 0.9, -n 4 -c 0.7, and -n 3 -c 0.5 for 90%, 70% and 50% respectively. UCLUST (version 7.0.1090) was run with --id 0.9, 0.7, 0.5, and Linclust (commit fe2369c) was executed using --target-cov and 0.9 --min-seq-id 0.9 or --min-seq-id 0.7 or --min-seq-id 0.5 for 90%, 70% and 50% respectively.

Clustering metagenomic sequences. We downloaded ∼1800 metagenomic and ∼400 metatranscriptomic datasets with assembled contigs from IMG/M [2] and NCBI's Sequence Read Archive [11] (ftp://ftp.ncbi.nlm.nih.gov/sra/wgs_aux) using the script metadownload.sh from https://bitbucket.org/martin_steinegger/linclust-analysis. We predicted genes and protein sequences using Prodigal [19] in the contigs and added the 40.2 million protein sequences from the Ocean Microbiome Reference Gene Catalog (OM-RGC) [10] and the 123 million sequences from UniParc [12]. Since many of the predicted protein sequences are fragments, we chose as acceptance criteria in the Hamming stage 2 a minimum coverage *of the shorter of the two sequences* of 99% and a sequence identity of 95% (Linclust options --target-cov 99 --min-seq-id 0.5). Running stage 2 but choosing the greedy clustering algorithm instead of greedy set-cover (Linclust options --cluster-mode 2) produced the set Metaclust95. We further clustered the sequences in Metaclust95 down to 50% with a minimum symmetrical coverage threshold of 80% using stages 3 to 5.

Metaclust95 and Metaclust50 protein sequence sets. These datasets are freely available as FASTA formatted flat files at https://metaclust.mmseqs.com/. We used the summarizeheaders tool [20] to generate headers for the Metaclust50 sequences that list the identifiers of all Metaclust95 member sequences.

Code availability. Linclust has been integrated into our free GPLv3-licenced MMseqs2 software suite [16]. The source code and binaries for Linclust can be download at https://github.com/soedinglab/mmseqs2.

Data availability. All scripts and benchmark data including command-line parameters necessary to reproduce the benchmark and analysis results presented here are available at https://bitbucket.org/martin_steinegger/linclust-analysis.