Fast and sensitive taxonomic classification for metagenomics with Kaiju

Menzel, Peter; Ng, Kim Lee; Krogh, Anders

Published in:
Nature Communications

DOI:
10.1038/ncomms11257

Publication date:
2016

Document version
Publisher's PDF, also known as Version of record

nature COMMUNICATIONS

ARTICLE

Received 26 Nov 2015 | Accepted 7 Mar 2016 | Published 13 Apr 2016

DOI: 10.1038/ncomms11257

OPEN

Fast and sensitive taxonomic classification for metagenomics with Kaiju

Peter Menzel[1], Kim Lee Ng[1] & Anders Krogh[1]

Metagenomics emerged as an important field of research not only in microbial ecology but also for human health and disease, and metagenomic studies are performed on increasingly larger scales. While recent taxonomic classification programs achieve high speed by comparing genomic *k*-mers, they often lack sensitivity for overcoming evolutionary divergence, so that large fractions of the metagenomic reads remain unclassified. Here we present the novel metagenome classifier Kaiju, which finds maximum (in-)exact matches on the protein-level using the Burrows–Wheeler transform. We show in a genome exclusion benchmark that Kaiju classifies reads with higher sensitivity and similar precision compared with current *k*-mer-based classifiers, especially in genera that are underrepresented in reference databases. We also demonstrate that Kaiju classifies up to 10 times more reads in real metagenomes. Kaiju can process millions of reads per minute and can run on a standard PC. Source code and web server are available at http://kaiju.binf.ku.dk.

[1] Department of Biology, University of Copenhagen, Copenhagen 2200, Denmark. Correspondence and requests for materials should be addressed to A.K. (email: krogh@binf.ku.dk).

Using random DNA shotgun sequencing, it is possible to directly obtain total genomic DNA from an environmental sample without the need for laboratory cultures. This 'metagenomic' approach has become a standard method for characterizing the biodiversity, gene contents and metabolic processes of bacterial and archaeal communities and is used on increasingly larger scales[1–3]. Due to decreasing cost of high-throughput sequencing (HTS) and the recent revelations of the importance of microbiomes for health and disease[4,5], metagenomic analyses are also likely to become part of routine clinical diagnostics and detection of pathogens.

One of the major biological questions in metagenomics is the inference of the composition of a microbial community, that is, the relative abundances of the sampled organisms. One approach is the assembly of metagenomic reads into contigs, which are then compared with reference genomes. However, assembly-free taxonomic classification is faster and more straightforward, because metagenomic assembly and quantification of taxon abundances from contigs pose additional computational challenges. Hence, given random shotgun sequencing reads, the underlying algorithmic problem is the assignment of individual reads to taxa, usually by comparison to a reference database. Traditionally, this task is solved by local sequence alignment either on nucleotide-level, when comparing sequencing reads with a database of microbial genomes, or on protein-level when translating reads to amino acid sequences and comparing with a catalogue of microbial genes. However, with increasing volumes of microbial genome databases and sequencing output, computational methods need to catch up, as traditional methods based on local sequence alignment are too slow to cope with the increasing amount of data.

For the similar problem of mapping sequencing reads to a reference genome, heuristic methods achieve speed improvements of orders of magnitude by using advanced index structures for fast identification and extension of short exact matches (seeds) between query and the reference genome[6]. However, these mappers are not suited for classification of metagenomic sequences, because they only work on DNA in a usually semi-global alignment model and assume near-identity of read sequences and reference genome.

Thus, programs have been developed for fast taxonomic classification of individual sequencing reads by using hash-based index structures built from a set of reference sequences, typically a database of complete microbial genomes. To achieve high speed, these algorithms do not use traditional local alignment methods, but rely on the identification of *k*-mers, short exact matching substrings of fixed-length *k*, in order to compare two nucleotide sequences. For the taxonomic assignment of reads, these programs typically preprocess the reference genomes by extracting all contained *k*-mers and storing them in the index for fast lookup. Then, the *k*-mers contained in each sequencing read are searched in this index and the read is assigned to a taxon based on the matching genomes. Recent programs following this paradigm are LMAT[7], Kraken[8] and Clark[9]. For example, Kraken builds an index from all *k*-mers found in the reference genomes and assigns each *k*-mer to the least common ancestor (LCA) of all species having that *k*-mer. Then, during the search, Kraken matches the *k*-mers found in the reads to this index and eventually assigns the read to the taxon with most matching *k*-mers by following a path from the root of the tree. Clark on the other hand only uses discriminative *k*-mers between sets of reference genomes belonging to a pre-defined taxonomic rank, for example, genus, which are then used to classify reads to a node in the taxonomic tree at that particular rank. While this approach reduces the size of the index, it, however, prohibits the assignment of reads to higher taxonomic levels in case of ambiguity and therefore requires the user to build different indices for each rank in the taxonomic tree. Genomic *k*-mers are also used in the LSA program[10], which can quickly sort metagenomic reads into bins for each species to aid the assembly of low-abundance species.

Fast classifiers using *k*-mers have so far been restricted to classification at the DNA level, where the fundamental requirement is a high sequence identity between reads and the reference database, so that in the minimal case at least one *k*-mer per read can be found in the database. Therefore, these methods work best for samples in which the majority of the species have been previously sequenced and their genomes are contained in the reference database and when a classification at the lowest possible level in the taxonomy is of importance. However in many samples, no reference genomes are available for a large fraction of the organisms.

Another general problem with metagenomic sequence comparison is a sampling bias in the phylogenetic distribution of available reference genomes. On the one hand, certain model organisms or pathogens, for example, from human microbiomes, are primary targets for microbial research and are therefore over-represented in the genome databases. On the other hand, species that were not possible to culture in the laboratory are underrepresented, which is a further challenge for the taxonomic classification of environmental samples, especially from extreme environments. In addition, the rate of evolution is faster for microbes and especially for viruses compared with eukaryotes due to higher replication rates. Thus, metagenomic studies continuously find novel habitats where large fractions of the sequence data remain unclassified or only show low sequence similarities to the known species[11,12].

In such samples using protein-level classification can increase accuracy, because protein sequences are more conserved than the underlying DNA, and microbial and viral genomes are typically densely packed with protein-coding genes[13,14]. In addition, protein sequence comparison is more tolerant to sequencing errors due to the degeneracy of the genetic code. Thus, there is a need for fast metagenome classifiers that are able to detect evolutionary distant relatives of the species with reference genomes based on amino acid sequence comparison. Several programs exist for seed-based local alignment of protein sequences, like BlastP and BlastX[15], or the faster methods using index structures, like RapSearch[16] and Diamond[17]. However, these alignment programs are generally slower than the *k*-mer-based methods, and they report all alignments to the reference database, which need to be analysed further for taxonomic classification[18].

Here we present Kaiju, a novel program for fast taxonomic classification based on sequence comparison to a reference database of microbial proteins. We show that our approach is able to classify more reads in real metagenomic data sets and evaluate its performance in a benchmark study, which simulates the classification of a novel genome taking the sampling bias of reference databases into account.

Results

Protein-level sequence classification. Kaiju translates metagenomic sequencing reads into the six possible reading frames and searches for maximum exact matches (MEMs) of amino acid sequences in a given database of annotated proteins from microbial reference genomes. If matches to one or more database sequences are found for a read, Kaiju outputs the taxonomic identifier of the corresponding taxon, or it determines the LCA in the case of equally good matches to different taxa. Kaiju's underlying sequence comparison algorithm uses the

Burrows–Wheeler transform (BWT) of the protein database, which enables exact string matching in time proportional to the length of the query, to achieve a high classification speed.

In k-mer-based methods, the size of k governs the sensitivity and precision of the search. If k is chosen too large, no identical k-mers between read and database might be found, especially for short or erroneous reads, as well as for evolutionary distant sequences. If k is chosen too small, more false positive matches will be found. Therefore, in order to not be restricted by a prespecified k-mer size, Kaiju finds MEMs between reads and database to achieve both a high sensitivity and precision. Reads are directly assigned to a species or strain, or in case of ambiguity, to higher level nodes in the taxonomic tree. For example, if a read contains an amino acid sequence that is identical in two different species of the same genus then the read will be classified to this genus. Kaiju also offers the possibility to extend matches by allowing a certain number of amino acid substitutions at the end of an exact match in a greedy heuristic approach using the BLOSUM62 substitution matrix. See the Methods section for a detailed description of Kaiju's algorithm.

Genome exclusion benchmark. Benchmarking a classifier's accuracy can be done by simulation studies, which, knowing the ground truth about the origin of the simulated reads, can assess the sensitivity and precision of the classification. However, the benchmark protocol needs to reflect the real obstacles in metagenomic studies, which do not only include the bias and errors of the sequencing technology, but also the microbial composition of the sample at hand. Thus, we devised a simulation benchmark, which emulates the often limited availability of reference genomes and its impact on the classification performance when faced with a novel strain or species found in the metagenomic sample. To this end, we created a reference database of 2,724 bacterial and archaeal genomes and selected the subset of genomes belonging to genera that have at least 2 and most 10 genomes in the database. For each of the 882 genomes in this subset, we simulated 4 sets of Illumina and 1 set of Roche/454 sequencing reads and created a version of the reference database excluding that genome. This stripped reference (now containing 2,723 genomes) was then used to classify the simulated reads and we measured the number of classified reads, sensitivity and precision on genus, as well as phylum-level (see Methods). The number of genomes per genus serves as an indicator for the difficulty of the classification problem. For example, it is much harder to assign a novel genome to its genus when there is only one other genome of the same genus already available in the database. On the other hand, if there are 10 genomes available in a genus, it is typically much easier to classify the reads from the excluded genome to its genus with 9 remaining genomes available.

We compared the performance of Kaiju with the two k-mer-based programs Kraken and Clark, which performed best in speed and accuracy in a recent benchmark study[18]. Both Kraken and Clark use a reference database comprising whole genomes and construct an index of the contained nucleotide k-mers. While Kraken uses a default length of $k=31$, the user can chose k in Clark during database construction and values of $k=20$ and $k=31$ are recommended for highest sensitivity and highest precision, respectively. Therefore we chose values of $k=20$ and $k=31$ in Clark to illustrate the influence of the choice of k on the classification performance. Kaiju was run in the fastest MEM mode (with minimum fragment length $m=11$), as well as in the heuristic Greedy mode (with minimum score $s=65$), allowing either only one (Greedy-1) or up to five (Greedy-5) amino acid substitutions during the search.

Genomes were binned into categories in the range 2–10 according to the total number of genomes in the genus. Sensitivity and precision were calculated as the mean across all genomes in each category for each program and the five different types of simulated reads. Figure 1 compares the genus-level sensitivity and precision and Supplementary Fig. 1 shows the mean percentage of classification attempts for each genus category.

As expected, all programs have the lowest percentage of classified reads and the lowest sensitivity in those genera with only few available genomes and highest sensitivity in genera with seven or more genomes. Second, the read length is a major determinant for sensitivity as there is a much higher chance of finding a matching sequence to the reference database with increasing read length. Especially Kaiju gains a further increase of sensitivity from longer reads, as the chance of an overlap to a protein-coding region additionally increases with read length. For example, when looking at the Illumina single-end 100 nt reads, Greedy-5 achieves the highest sensitivity of 29% of all programs in genera with only two genomes, whereas Clark-k31 has the lowest sensitivity of 16%. In contrast for Illumina paired-end 250 nt reads, Greedy-5 achieves 59% sensitivity, whereas Clark-k31 only achieves 36%. With increasing number of genomes per genus, the difference between Greedy-5 and both Clark and Kraken shrinks to a few per cent, as the chance of finding at least one k-mer per read increases with more available reference genomes. Kaiju's MEM mode has lower sensitivity compared with Greedy modes in all cases, because it only searches for exact matches, which is especially visible in short reads.

Similarly, the precision of all programs is lowest in genera with only two genomes and increases with higher number of available genomes. However, the differences between the programs is much smaller compared with sensitivity, with Clark-k31 showing the highest precision by a small margin in most cases. When comparing Clark-k31 and Kraken, Clark has consistently a little bit lower sensitivity and a bit higher precision than Kraken. The difference between Clark-k20 and Clark-k31 nicely illustrates the trade-off between sensitivity and precision depending on the k-mer size. However, the loss in precision is generally higher than the gain in sensitivity when using $k=20$.

Supplementary Fig. 2 shows the phylum-level sensitivity and precision. At this level, the difference in sensitivity between Kaiju and Kraken is generally higher, because more reads are assigned to ranks higher than genus by Kaiju's LCA algorithm, whereas Kraken's weighted path algorithm usually assigns reads to the lowest possible level. Again, the increase in sensitivity with increasing read length is higher in Kaiju compared with Kraken and Clark. For example, in genera with only two genomes, Greedy-5 achieves between 41% (Illumina single-end 100 nt) and 84% (Illumina paired-end 250 nt), whereas Clark achieves between 17 and 44%. On phylum-level, all modes of Kaiju achieve $\sim 10\%$ higher sensitivity than Clark and Kraken up to the highest category with genera containing 10 genomes.

Phylum-level precision is generally much higher ($>90\%$) for all methods and all read types compared with genus-level, because the chance of false positive matches outside the correct phylum is lower. Again, Clark-k20 consistently yields a much lower precision compared with Clark-k31 and the other programs, however, it also gains more sensitivity on phylum-level classification compared with genus-level. This can be attributed to the removal of k-mers that are shared across genera for the genus-level classification, which, however, can be used on the phylum-level.

Figure 2 shows the mean genus-level and phylum-level sensitivity and precision across all 882 measured genomes for

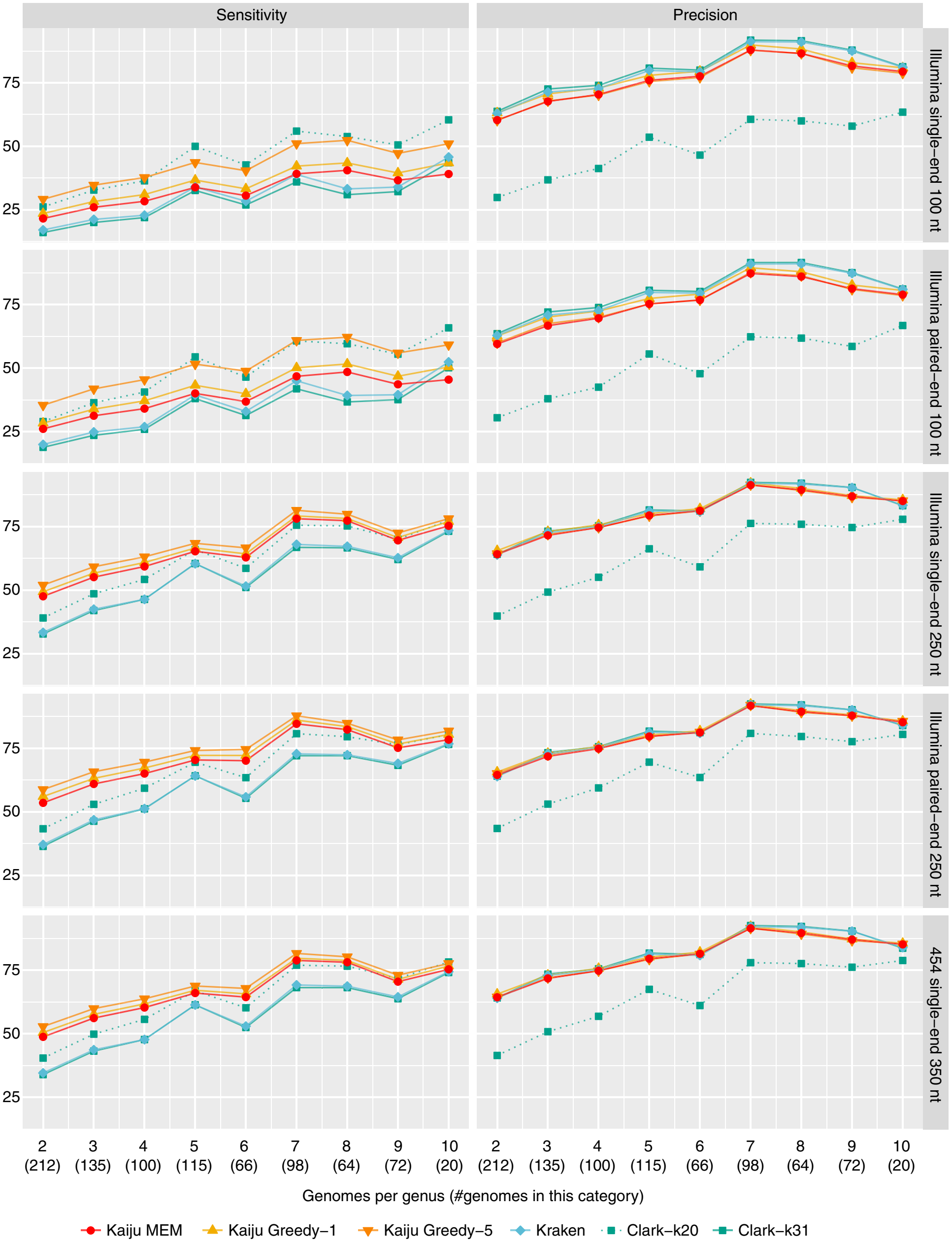

Figure 1 | Genus-level sensitivity and precision. Sensitivity and precision are shown as average for each bin of genera for the five different types of reads and the three programs. The *x*-axis denotes the number of genomes in the genus and the total number of genomes in that category. For example, 212 of the measured 882 genomes belong to the 106 genera with only 2 available genomes, and the data points show the mean sensitivity and precision across all 212 genomes in that category. Kaiju was run in MEM mode with length threshold $m = 11$ a.a. and in Greedy mode with either 1 or up to 5 allowed mismatches and a score threshold $s = 65$. Kraken uses $k = 31$ and Clark was run with both $k = 31$ and $k = 20$, which is denoted by the dotted line.

the five different read types. The biggest gap for sensitivity and precision between the read types occurs for all programs between both paired-end and single-end 100 nt and the single-end 250 nt Illumina reads. Highest sensitivity is achieved by Greedy-5, followed by Greedy-1, MEM, Kraken and Clark in the paired-end 250 nt reads. Precision is highest for Clark closely followed by Kraken both on genus-level and phylum-level. Especially in the 100 nt reads, Kaiju's precision is lower, but the gain in sensitivity remains higher than the loss in precision. For the 250 nt reads and longer, Kaiju's precision is marginally lower than Kraken and Clark-k31, while sensitivity is much higher.

In this analysis, we used cutoff values of minimum required match length $m = 11$ in Kaiju's MEM mode and minimum required match score $s = 65$ in the Greedy modes. These parameters govern the accuracy of the classification, similar to the choice of k in the k-mer-based classifiers. Thus, we also examined Kaiju's accuracy using different values for m and s. Supplementary Figure 3 shows the trade-off between sensitivity and precision of the classification depending on the choice of m or s. Similar to the choice of k in Clark, the sensitivity is highest and precision is lowest for small cutoff values. Increasing the cutoffs results in lower sensitivity but higher precision. However, the increase in sensitivity between $m = 11$ and $m = 12$ is higher than the loss in precision in all data sets both on genus-level, as well as phylum-level. Similarly in the Greedy modes, $s = 65$ also yields higher gain in sensitivity than loss in precision.

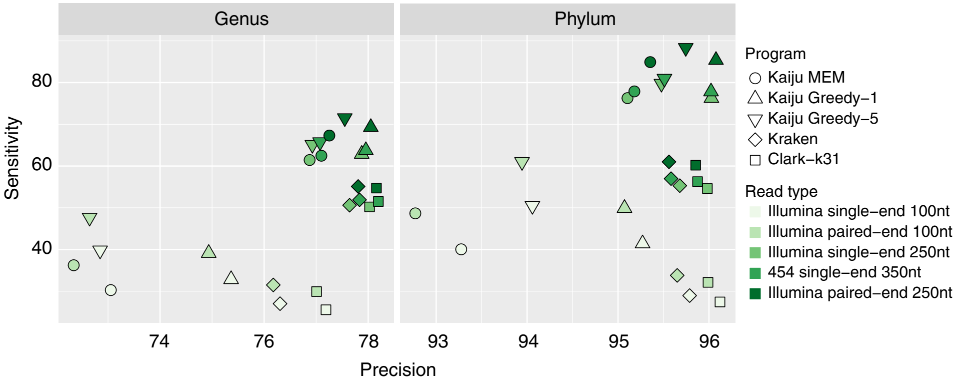

Figure 2 | Average sensitivity and precision. For each of the five types of reads, sensitivity and precision were averaged over all 882 measured genomes in the benchmark, showing the overall performance of each program.

	MEM	Greedy	Kraken	Merged
Human vagina	58.6	63.3	45.5	66.4
Human saliva	56.1	61.8	37.4	64.6
Human gut	51.6	56.7	34.7	61.1
Cat gut	51.8	58.9	23.0	63.1
Lake	26.5	33.7	3.1	34.0
River plume	31.8	38.8	5.8	39.3
Baltic seawater	25.3	33.9	5.6	34.8
Desert soil	19.3	24.4	3.8	24.7
Bioreactor sediment	65.0	72.8	16.2	73.1
Bioreactor compost	50.3	58.5	15.1	59.5
Mean	43.6	50.3	19.0	52.1

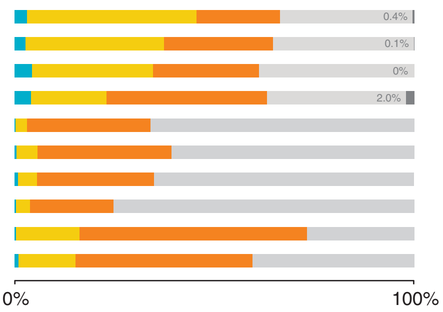

Figure 3 | Classification of real metagenomes. Percentage of classified reads in 10 real metagenomes for Kaiju MEM ($m = 12$) and Greedy-5 ($s = 70$), as well as Kraken ($k = 31$). The Merged column shows the percentage of reads that are classified by at least one of Greedy-5 or Kraken. The Venn-Bar-diagram visualizes the percentage of reads that are classified either only by Kraken (blue), Greedy-5 (orange) or both (yellow). Grey bars in the human and cat samples denote the percentage of reads mapped to the respective host genomes.

Real metagenomes. To assess how many reads can actually be classified in real metagenomic data sets, we arbitrarily selected 10 previously published data sets from different microbiomes that were sequenced using various different HTS instruments. The two data sets from human saliva and vagina samples were already used by Ounit *et al.*[9]. The other eight samples are from human and cat gut, a freshwater lake, the Amazon river plume and Baltic sea, xeric desert soil and from two bioreactors that were inoculated with microbes from Wadden Sea sediment and compost environments. Supplementary Table 1 lists metadata and accession numbers for the data sets. The same database comprising 2,724 genomes from our exclusion benchmark serves as a reference database. We classified the 10 data sets using Kraken ($k = 31$) and Kaiju in MEM and Greedy-5 modes with more conservative cutoff values of $m = 12$ and $s = 70$, respectively, which showed on average a similar precision as Kraken across the five types of reads in our exclusion benchmark, see Supplementary Fig. 3. We also mapped the four human and cat samples to their respective host genomes using BWA[19], and the percentage of mapped reads was at most 2%.

Figure 3 shows the percentage of classified reads from each data set for MEM, Greedy-5 and Kraken, as well as the overlap and combined percentage of Greedy-5 and Kraken. Generally, Kaiju's MEM mode classifies between 13.1% (Human vagina) and 48.8% (Bioreactor sediment) more reads than Kraken, which is further increased to 17.8 and 56.6% in Kaiju's Greedy-5 mode. The percentages of reads that are classified by Kraken, but unclassified by Greedy-5 range between 0.3% (Desert soil and Lake) and 4.4% (Human gut). Across all data sets, the number of reads that were classified by both Greedy-5 and Kraken (overlap) varies between 2.8% (Lake) and 42.4% (Human vagina). By merging the results from Greedy-5 and Kraken, between 24.7% (Desert soil) and 73.1% (Bioreactor sediment) of the total reads can be classified.

As expected, the environmental samples, especially from the extreme xeric desert, but also the aquatic microbiomes, pose the biggest challenges for taxonomic assignment. In those samples Kaiju's protein-level comparison with substitutions allows for a more sensitive sequence comparison resulting in more classified reads. However, even in the human microbiomes Kaiju's protein-level classification adds >20% additional classified reads to Kraken's result.

We also run each data set through Clark ($k = 31$) with its phylum-level database and it classified fewer reads than Kraken in all cases (data not shown). In principle, if only a small fraction of reads were classified, classification could be done using a smaller k-mer size in Kraken and Clark or smaller cutoff values in Kaiju to increase the number of classified reads. The trade-off, however, would be a decreased precision as shown in our benchmark and also discussed in the study by Ounit *et al.*[9].

HiSeq and MiSeq mock communities. In addition to the real metagenomes, we also measured Kaiju's and Kraken's performance using the same metrics and reference database on the HiSeq and MiSeq mock community data sets from previous benchmarks[8,9]. They comprise 10k real sequencing reads from 10 bacterial strains with mean read length of 92 nt (HiSeq) and 156 nt (MiSeq). All strains belong to genera that are associated with human microbiomes or human pathogens and have typically many reference genomes available. Supplementary Table 2 shows sensitivity and precision on both genus- and phylum-level of Kaiju in Greedy-5 mode and Kraken ($k=31$) using the same reference database as above. In the HiSeq data set, Kaiju has 73.3% sensitivity (Kraken: 78.0%) and 94.4% precision (Kraken: 99.2%) on genus-level, and 78.1% sensitivity (Kraken: 78.8%) and 98.3% precision (Kraken: 99.7%) on phylum-level. Because the short reads can only yield short amino acid fragments that are more likely found across genera, many reads are assigned to higher ranks resulting in a lower genus-level sensitivity. In addition, the short read length results in generally lower overlap with protein-coding regions and therefore Kraken yields a higher sensitivity, because it can classify those. In the MiSeq data set, the difference between both programs on genus-level is similar, whereas Greedy-5 yields 8% higher sensitivity and 1% higher precision on phylum-level compared with Kraken.

Runtime and memory. The read data set for the runtime benchmark contained 27.24 m reads comprised of 10k randomly sampled reads from each of the 2,724 genomes in our accuracy benchmark, which served again as the reference database. For the five different types of reads, the classification speed of Clark and Kraken using $k=31$ and of Kaiju's modes MEM, Greedy-1 and Greedy-5 was measured using 25 parallel threads (see Methods section for specification of the hardware).

Figure 4 shows the number of processed reads per second (r.p.s.). The classification of the short single-end 100 nt reads is the fastest in all programs (Kaiju MEM: 173k r.p.s., Kraken: 165k r.p.s., Clark: 93k r.p.s.), whereas classification of the paired-end 250 nt (MEM: 97k r.p.s., Kraken: 24k r.p.s., Clark: 19k r.p.s.) takes the longest time. In the long reads, Kaiju can benefit from search space pruning by finding long MEMs first, whereas Kraken and Clark have to analyse more k-mers compared with the shorter reads. Naturally, Kaiju's MEM mode is much faster than the Greedy modes, which extend the search space and also need to calculate the scores for each match. Depending on the read type, Greedy-5 classifies between 36k and 76k r.p.s. Greedy-1 with only one allowed mismatch is faster than Greedy-5 and can classify between 54k and 100k r.p.s.. Interestingly, Kaiju's Greedy mode is faster in longer and paired-end reads compared with the 100 nt reads. This is due to the pruning of the search space by discarding query sequences that cannot achieve higher scores than the best scoring match, which is usually found earlier in longer and paired-end reads (see Methods). While Kaiju MEM is the fastest program in most cases, especially for the short reads, Greedy-5 generally takes the longest time, nicely demonstrating the trade-off between speed and sensitivity.

The measured peak memory consumption during the classification is 5.6 GB for Kaiju, 72 GB for Kraken and between 65 and 78 GB for Clark depending on the read length.

The construction of Kaiju's index from the protein sequence database takes 8 min with peak memory usage of 24 GB using 25 threads (Kraken: 1h26m/165 GB, Clark: 3h57m/152 GB). The memory requirement is largely determined by the number of parallel threads for sorting the suffix array, which can, for example, be reduced to only 6.6 GB with only five threads. Kaiju's final index size on disk is 4.8 GB (Kraken: 73 GB, Clark: 39 GB).

Discussion

When performing sequence comparison, as in the case of taxonomic assignment using a reference database, there is the obvious trade-off between an algorithm's speed and accuracy. While the traditional local alignment would return optimal alignments, its slow runtime prohibits its use on large HTS data sets. On the other hand, while k-mer-based methods are very fast, they often lack sensitivity and a big fraction of the metagenomic reads might remain unclassified, as can be seen from Fig. 3. Kaiju therefore uses MEMs (with optional substitutions) on the protein-level instead of nucleotide-level to increase sensitivity of the classification while maintaining a high precision. By using the BWT as an index for the reference protein database, Kaiju is fast enough for classifying up to millions of reads per minute, depending on the read length and the number of allowed mismatches, and is typically faster than the two k-mer-based methods Kraken and Clark. In addition, the memory-efficient implementation of the FM-index and suffix array make Kaiju's memory footprint small enough (below 6 GB on our benchmark database) for running it on a standard PC. Kaiju does not have a limit on the input sequence length, so it could in principle also classify assembled contigs.

The aim of using protein-level sequence comparison is to improve the classification of metagenomes comprising species that are evolutionary distant to the species in the reference or belong to genera that have only few reference genomes available. Therefore, we focused on those genera with ≤10 genomes in our

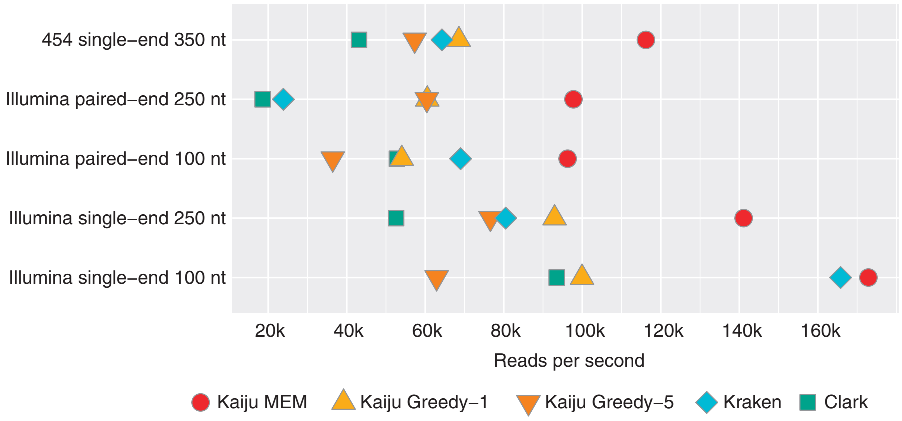

Figure 4 | Classification speed. Performance was measured in processed reads per second for each program using 25 parallel threads for classifying a set of 27.24 m simulated reads for the five different read types.

genome exclusion benchmark, because the classification problem becomes easier once many reference genomes are available and can be mostly accomplished by nucleotide-level sequence comparison, which is also better suited for strain typing because of the finer resolution regarding SNPs. Our benchmark on 882 genomes and five types of simulated reads shows that Kaiju consistently achieves a much higher sensitivity with only little loss of precision compared with Kraken and Clark, which use fixed-length k-mers. The difference was especially visible in genera with only few available genomes. Contrary to classification based on marker genes, for example, 16S ribosomal RNA, a genus-level classification of shotgun metagenomic reads is impossible for genera with no available reference genomes. However, they might still be classified to their correct family, if genomes from other genera in this family are available.

The obvious disadvantage of protein-level sequence classification is the inability to classify reads originating from non-protein-coding genomic regions. Especially when the genomes of the sequenced microbial strains are also contained in the reference database, Kaiju would be less sensitive than nucleotide-level classifiers, which can assess the entire genome, as seen in the HiSeq and MiSeq data sets. However, due to the high density of protein-coding genes in microbial genomes, the probability of overlap between individual sequencing reads and protein-coding genes increases substantially with increasing read lengths. Furthermore, when using paired-end sequencing, the chance of one mate overlapping with a protein-coding gene is higher than for single-end sequencing, which was also shown in our benchmark where longer and paired-end reads had higher sensitivity compared with shorter single-end reads.

In our set of 10 randomly selected real metagenomic data sets, Kaiju classifies on average twice as many reads as Kraken. The highest differences are observed in samples from non-human microbiomes, showing that especially the classification of environmental samples with high evolutionary distances to the reference genomes can gain from Kaiju's more sensitive sequence comparison. By combining Kaiju's and Kraken's output, between 24 and 73% of reads can be classified across the various samples.

Principally, Kaiju's algorithm is not limited to assigning reads to taxa, but can also be used for fast searching in arbitrary protein databases, for example, when querying novel bacterial genomes against a database of resistance genes or a collection of proteins with functional annotation. The certainly expected increase of reference database volumes in the coming years can easily be handled by Kaiju, due to the usage of memory-efficient index structures.

Methods

Metagenome classifier. Kaiju classifies individual metagenomic reads using a reference database comprising the annotated protein-coding genes of a set of microbial genomes. We employ a search strategy, which finds maximal exact matching substrings between query and database using a modified version of the backwards search algorithm in the BWT[20,21]. The BWT[22] is a text transformation that converts the reference sequence database into an easily searchable representation, which allows for exact string matching between a query sequence and the database in time proportional to the length of the query. While in the context of read mapping, MEMs have been used as a fast method for identifying seeds of mapping regions in the reference genome, for example, in (refs 23,24), we use MEMs to quickly find those sequences in the reference database that share the longest possible subsequence with the query. Backtracking through the BWT can be sped up by using a lookup table for occurrence counts of each alphabet letter, which was first proposed by Ferragina and Manzini[20] and is often called FM-index. Kaiju employs a sparse representation of this table by using checkpoints, which allows for decreasing the otherwise large memory requirement due to the size of the amino acid alphabet. The initial suffix array used for calculating the BWT is also implemented as a sparse suffix array with adjustable size, which further reduces the index size with only little impact on runtime, because the suffix array is only needed for determining the name of the database sequence once the best match for a read is found. Thus, Kaiju is the first program to efficiently use the BWT and FM-index on a large protein database, allowing querying large sets of sequencing reads.

Figure 5 illustrates the steps in Kaiju's algorithm: First, Kaiju translates each read into the six possible reading frames, which are then split at stop codons into amino acid fragments. These fragments are sorted by length, and, beginning with the longest fragment, queried against the reference database using the backwards search in the BWT. Given a query fragment of length n and the minimum required match length m, the backwards search is started from all positions between n and $n-m$ in the query and the longest MEM is retained. If one or more matches of length $l>m$ are found, m is set to l and the next fragment in the ordered list is queried against the database if its length is at least l, otherwise the search stops. Once the search is finished and one or more matches are found, the taxon identifier from the corresponding database sequence is retrieved from the suffix array and printed to the output. If equally long matches are found in multiple taxa, Kaiju determines their LCA from the taxonomic tree (Supplementary Fig. 6) and outputs its taxon identifier. Thus, each read is always classified to the lowest possible taxonomic level given the ambiguity of the search result.

The minimum required MEM length m is the major parameter for trading sensitivity versus precision (with little impact on runtime). If the error rate e of the sequencing reads is known and the evolutionary distance between reference genome and sequenced genome is negligible, m can be estimated from e and the read length[23]. However, in metagenomics, the evolutionary distance, which adds variation on top of sequencing errors, is not known *a priori*. At least, one can

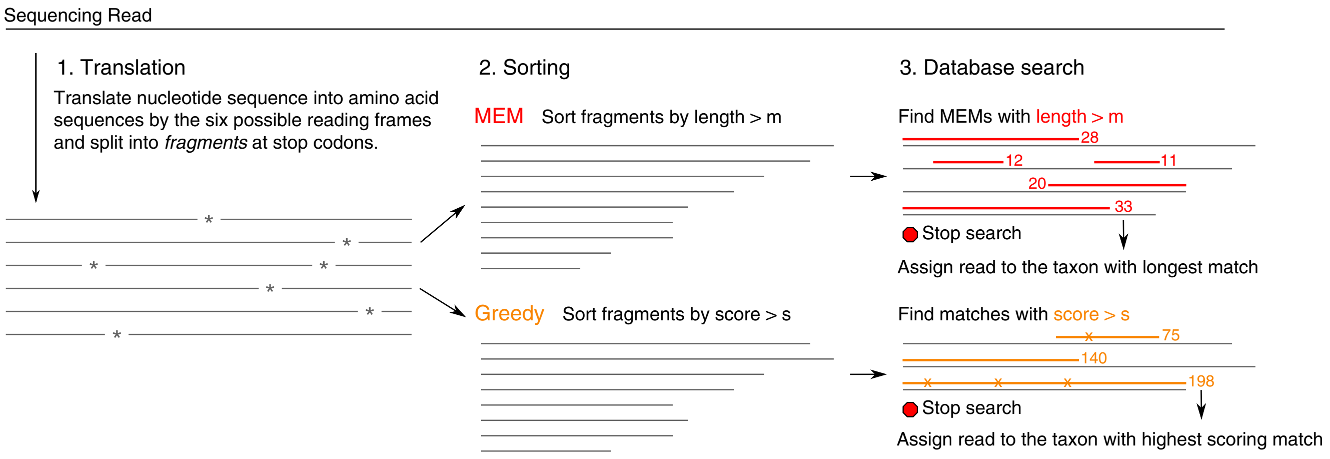
Sequencing Read
1. Translation
Translate nucleotide sequence into amino acid sequences by the six possible reading frames and split into *fragments* at stop codons.
2. Sorting
MEM
Sort fragments by length > m
Greedy
Sort fragments by score > s
3. Database search
Find MEMs with length > m
28
12
11
20
33
Stop search
Assign read to the taxon with longest match
Find matches with score > s
75
140
198
Stop search
Assign read to the taxon with highest scoring match

Figure 5 | Kaiju's algorithm. First, a sequencing read is translated into the six possible reading frames and the resulting amino acid sequences are split into fragments at stop codons. Fragments are then sorted either by their length (MEM mode) or by their BLOSUM62 score (Greedy mode). This sorted list of fragments is then searched against the reference protein database using the backwards search algorithm on the BWT. While MEM mode only allows exact matches, Greedy mode extends matches at their left end by allowing substitutions. Once the remaining fragments in the list are shorter than the best match obtained so far (MEM) or cannot achieve a better score (Greedy), the search stops and the taxon identifier of the corresponding database sequence is retrieved.

estimate the false positive rate by counting random matches. To this end, we created a shuffled version of the microbial subset of NCBI's NR protein database, using uShuffle[25] with a window length of 100 amino acids, and searched for MEMs between simulated metagenomic reads and the shuffled database. Supplementary Figure 5 shows the cumulative sum of matches to the shuffled database sorted by the length of the match, and one can observe that ~95% of them are ≤11 amino acids long. When classifying simulated reads against the original database, >75% of wrong classifications and only ~2% of correct classifications have length ≤11. We therefore chose $m=11$ as the default minimum match length in Kaiju.

Searching for MEMs is the fastest possible search strategy, but its sensitivity decreases with increasing evolutionary distance between query and target, where more and more amino acid substitutions occur and exact matches become shorter. Therefore, allowing for substitutions during the backwards search can bridge mismatches and extend the match at the cost of an exponential increase of runtime depending on the number of allowed mismatched positions. Because of the rapid expansion of the search space, especially with the 20 letter amino acid alphabet, one could employ a greedy heuristic, in which substitutions are only introduced at the end of a match instead of all positions in the query sequence. Therefore, we also implemented a Greedy search mode in Kaiju, which first locates all MEMs of a minimum seed length (default 7) and then extends them by allowing substitutions at the left ends of each seed match. From there, the backwards search continues until the next mismatch occurs. Eventually the search stops once the left end of the query is reached or if the maximum allowed number of substitutions has been reached.

Since amino acid substitutions in homologous sequences are non-uniform, a further speed-up can be gained by prioritizing the most likely substitutions at each position. By using an amino acid substitution model, a total score for each match can be calculated, as in standard sequence alignment, which is then used to rank multiple matches and select the taxon from the database for classification. Therefore, after the translation of a read into a set of amino acid fragments, we rank the fragments by their BLOSUM62 score and start the database search with the highest scoring fragment. For each substituted amino acid, the modified fragment is placed back into the search list according to its new (now lower) score. Once a match is found, which has a higher score than all remaining fragments in the search list and a score above the minimum score threshold s, the search stops and this highest scoring match is used for classifying the read. If multiple matches to several different database entries have the same score, Kaiju classifies the read to their LCA as above. Again, the minimum required score s necessary for avoiding random matches can be estimated by using a shuffled database and we chose $s=65$ as default value for Kaiju's Greedy mode.

Kaiju is implemented as a command-line program in C/C++ and is also available via a web server. Input files containing the (single-end or paired-end) reads can either be in FASTA or FASTQ format. Kaiju outputs one line for each read (or read pair), containing the read name and the NCBI taxon identifier of the assigned taxon, as well as the length or score of the match. Optionally, Kaiju can also produce a summary file with the number of reads assigned per taxon, which can be loaded into Krona[26] for interactive visualization. We also include a utility program that can merge the classification results from different runs or programs, for example, for merging Kaiju and Kraken results.

Performance evaluation. The primary goal of Kaiju's protein-level classification is to improve classification of those parts of a metagenome that are only distantly related to the known sequences or belong to a branch of the phylogeny that is underrepresented in the reference database. We therefore devised a benchmark study, which addresses this problem by simulating the classification of metagenomic reads from a novel strain or species that is not contained in the reference database.

For our benchmark data set, we downloaded a snapshot of all complete bacterial and archaeal genomes from the NCBI FTP server (date: 16 December 2014). Only those genomes were retained that are assigned to a species belonging to a genus and have a full chromosome with annotated proteins, resulting in a total of 2,724 genomes belonging to 692 distinct genera. Supplementary Fig. 4 shows the distribution of genomes to genera, illustrating the large variance in the number of sequenced genomes for each genus. For example, the genus *Streptococcus* contains 121 genomes, whereas 405 genera have only 1 available genome, 106 genera have 2 available genomes and so on. The distribution clearly illustrates a sampling bias and the sparseness across large parts of the phylogeny.

From the total of 2,724 genomes, we extracted those genera that have at least 2 and at most 10 genomes assigned. This resulted in a list of 242 genera comprising 882 genomes, for which we measured the classification performance individually. For each of the 882 genomes, we simulated five sets of HTS reads and created a reference database not containing this genome that is then used to classify the simulated reads. Reads were simulated from the whole genome (including plasmids) using ART[27]. The four sets of Illumina reads contain 50k reads of length either 100 or 250 nt, both in single-end and paired-end mode. Another set of 50k Roche/454 reads with minimum length of 50 nt and mean length of 350 nt was also simulated using ART.

To evaluate classification accuracy, we measured the number of classified reads, as well as sensitivity and precision on genus- and phylum-levels. Sensitivity was calculated as the percentage of reads assigned to the correct genus/phylum out of the total number of reads in the input. Precision was calculated as the percentage of reads assigned to the correct genus/phylum out of the number of classified reads, excluding reads classified correctly to a rank above genus/phylum-level. The same measurements were used in the study by Ounit *et al.*[9]. Kraken (v0.10.4b) and Clark (v1.1.3) were run in their default modes using $k=31$ for highest precision, and Clark was also run using $k=20$. Kaiju was run in MEM mode using minimum match lengths $m=11\ldots14$ and in Greedy-1 and Greedy-5 modes (allowing only 1 or up to 5 substitutions) using minimum match scores $s=55\ldots80$.

Speed measurements were run on an HP Apollo 6000 System ProLiant XL230a Gen9 Server, which has two 64-bit Intel Xeon E5-2683 2 GHz CPUs (14 cores each), 128 GB DDR4 memory and a 500 GB 7200 r.p.m. SATA disk (HP 614829-002). Kraken and Clark were run in default modes with $k=31$ and Kaiju was run in MEM ($m=12$), as well as Greedy-1 and Greedy-5 ($s=65$) modes with an index that uses a suffix array exponent of 3. Performance was measured in processed reads (or read pairs) per second (r.p.s.) using 25 parallel threads. While Kaiju and Clark need to preload their index into memory before the classification starts, Kraken can either preload the index or only load necessary segments during the classification. We therefore measured Kraken's speed using both options, and it turned out that Kraken runs faster without preloading on our hardware. We therefore report its performance without preloading. For each of the five types of simulated reads from our exclusion benchmark, we created a data set comprising 10k reads from each genome in the reference database, resulting in 27.24 m reads for each read type. Each combination of program and read type was measured four times to reduce impact of caching and I/O fluctuations and the fastest run of the replicates is reported.

Figures were made with the ggplot2 package[28] in the R statistical software[29].

References

1. Riesenfeld, C., Schloss, P. & Handelsman, J. Metagenomics: genomic analysis of microbial communities. *Annu. Rev. Genet.* **38,** 525–552 (2004).
2. Shokralla, S., Spall, J., Gibson, J. & Hajibabaei, M. Next-generation sequencing technologies for environmental DNA research. *Mol. Ecol.* **21,** 1794–1805 (2012).
3. Segata, N. *et al.* Computational meta'omics for microbial community studies. *Mol. Syst. Biol.* **9,** 666 (2013).
4. Kinross, J., von Roon, A., Holmes, E., Darzi, A. & Nicholson, J. The human gut microbiome: implications for future health care. *Curr. Gastroenterol. Rep.* **10,** 396–403 (2008).
5. Wade, W. The oral microbiome in health and disease. *Pharmacol. Res.* **69,** 137–143 (2013).
6. Fonseca, N., Rung, J., Brazma, A. & Marioni, J. Tools for mapping high-throughput sequencing data. *Bioinformatics* **28,** 3169–3177 (2012).
7. Ames, S. *et al.* Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics* **29,** 2253–2260 (2013).
8. Wood, D. & Salzberg, S. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* **15,** R46 (2014).
9. Ounit, R., Wanamaker, S., Close, T. & Lonardi, S. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics* 16, 236, 2015).
10. Cleary, B. *et al.* Detection of low-abundance bacterial strains in metagenomic data sets by eigengenome partitioning. *Nat. Biotechnol.* **33,** 1053–1060 (2015).
11. Menzel, P. *et al.* Comparative metagenomics of eight geographically remote terrestrial hot springs. *Microb. Ecol.* **70,** 411–424 (2015).
12. Sunagawa, S. *et al.* Structure and function of the global ocean microbiome. *Science* **348,** 1261359 (2015).
13. Bentley, S. & Parkhill, J. Comparative genomic structure of prokaryotes. *Annu. Rev. Genet.* **38,** 771–792 (2004).
14. Garrett, R. A. & Klenk, H.-P. (eds) *Archaea: Evolution, Physiology, and Molecular Biology* (Wiley-Blackwell, 2007).
15. Altschul, S., Gish, W., Miller, W., Myers, E. & Lipman, D. Basic local alignment search tool. *J. Mol. Biol.* **215,** 403–410 (1990).
16. Zhao, Y., Tang, H. & Ye, Y. RAPSearch2: a fast and memory-efficient protein similarity search tool for next-generation sequencing data. *Bioinformatics* **28,** 125–126 (2012).
17. Buchfink, B., Xie, C. & Huson, D. Fast and sensitive protein alignment using DIAMOND. *Nat. Methods* **12,** 59–60 (2015).
18. Lindgreen, S., Adair, K. & Gardner, P. An evaluation of the accuracy and speed of metagenome analysis tools. *Sci. Rep.* **6,** 19233 (2016).
19. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* **25,** 1754–1760 (2009).
20. Ferragina, P. & Manzini, G. in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. FOCS '00, 390-398 (IEEE Computer Society, 2000).
21. Frellsen, J., Menzel, P. & Krogh, A. in *Comprehensive Biomedical Physics* (ed. Brahme, A.) 41–50 (Elsevier, 2014).

22. Burrows, M. & Wheeler, D. *A Block-sorting Lossless Data Compression Algorithm*. SRC Research Report 124 (Digital Equipment Corporation, Palo Alto, California, USA, 1994).
23. Liu, Y. & Schmidt, B. Long read alignment based on maximal exact match seeds. *Bioinformatics* **28,** i318–i324 (2012).
24. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. Preprint at http://arxiv.org/abs/1303.3997 (2013).
25. Jiang, M., Anderson, J., Gillespie, J. & Mayne, M. ushuffle: a useful tool for shuffling biological sequences while preserving the k-let counts. *BMC Bioinformatics* **9,** 192 (2008).
26. Ondov, B., Bergman, N. & Phillippy, A. Interactive metagenomic visualization in a web browser. *BMC Bioinformatics* **12,** 385 (2011).
27. Huang, W., Li, L., Myers, J. & Marth, G. ART: a next-generation sequencing read simulator. *Bioinformatics* **28,** 593–594 (2012).
28. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis* (Springer-Verlag, 2009).
29. R Core Team. R. *A Language and Environment for Statistical Computing* (R Foundation for Statistical Computing, 2015).

Acknowledgements

The research leading to these results has received funding from the Novo Nordisk Foundation and the European Union 7th Framework Programme FP7/2007-2013 under grant agreement no. 265933. ELIXIR provided computational support through the Computerome.

Author contributions

P.M. and A.K. conceived the algorithm and developed the software together; P.M. and K.L.N. carried out the data analysis; P.M. and A.K. co-wrote the paper.

Additional information

Availability: Kaiju is written in C/C++ and the source code is freely available as open-source software under the GNU GPL3 at https://github.com/bioinformatics-centre/kaiju. A web server is available at http://kaiju.binf.ku.dk.

Supplementary Information accompanies this paper at http://www.nature.com/naturecommunications

Competing financial interests: The authors declare no competing financial interests.

Reprints and permission information is available online at http://npg.nature.com/reprintsandpermissions/

How to cite this article: Menzel, P. *et al.* Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nat. Commun.* 7:11257 doi: 10.1038/ncomms11257 (2016).