kASA: Taxonomic Analysis of Metagenomic Data on a Notebook

Silvio Weging[1,✉], **Andreas Gogol-Döring**[2], **and Ivo Grosse**[1,3]

[1]Institute of Computer Science, Martin Luther Universtity Halle-Wittenberg, Von-Seckendorff-Platz 1, 6120 Halle, Germany
[2]TH Mittelhessen University of Applied Sciences, Wiesenstraße 14, 35390 Giessen, Germany
[3]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Deutscher Platz 5e, 04103 Leipzig, Germany

The taxonomic analysis of sequencing data has become important in many areas of life sciences. However, currently available software tools for that purpose either consume large amounts of RAM or yield an insufficient quality of the results.

Here we present kASA, a *k*-mer based software capable of identifying and profiling metagenomic sequences with high computational efficiency and a small user-definable memory footprint. We ensure high sensitivity and precision via *k*-mers on amino acid level with a dynamic length of multiple *k*'s as well as custom algorithms and data structures that are optimised for external memory storage enable for the first time a full-scale metagenomics analysis without compromise on a standard notebook.

metagenomics | taxonomic identification | taxonomic profiling | *k*-mers | amino acids | third-generation sequencing

Correspondence: ***silvio.weging@informatik.uni-halle.de***

Introduction

Decoding the complex composition of microbial communities is important, for example, to determine the Earth's biodiversity or its role in human diseases. However, this poses a major challenge. One of the biggest problems is the sheer number of different organisms living together in a biotope. Most organisms thrive only in the community of other organisms so specific cultivation of single species is usually impossible. Microbial communities therefore must be analysed as a whole which is done by examining the totality of the genetic material in the community: the metagenome (1). The technology of Next-Generation Sequencing (NGS) offers the possibility of creating vast amounts of DNA sequences (so called 'reads') from the metagenome, usually several million per dataset, each with a length between tens and hundreds of nucleotides (2). The processing and taxonomic identification of these reads requires special bioinformatics methods.

In recent years, several software tools that are capable of comparing reads to a database containing genomic sequences of known organisms have been published (3). Apart from the investigation of microbial communities, most of these tools can also be used to detect and quantify contaminations in any sequencing data, so they can be extremely valuable for data quality assurance in most application areas of NGS.

One of the best known programs for comparing sequences to large databases is MegaBLAST (4), which uses a seed-and-extend heuristic for finding local alignments between a read and the database. While this tool has a high usability and accuracy, it is not very well suited for quickly processing large numbers of sequences, since the analysis of complete NGS datasets using MegaBLAST would require high computational effort. Other tools like Kraken (5), Kraken2 (5, 6) and Clark (7) use a more time-efficient approach called *k-mer sampling* that compares short sub-sequences of a fixed length k (k-mers) taken from the reads with a pre-computed index derived from the database. Kraken uses a least-common-ancestor (LCA) approach to infer taxonomic memberships, whereas Clark needs a fixed taxonomic rank beforehand. These methods allow very fast identification but require huge amounts of random-access-memory (RAM), sometimes more than 100 GB, especially for comprehensive databases. Other solutions try to balance between time, accuracy and memory consumption like Centrifuge (8), mash (9), sourmash (10), MetaCache (11), Kaiju (12) and many more. Unfortunately, even with those optimisations they still require large amounts of primary memory when using large comprehensive databases like NCBIs nucleotide sequence database (13). This means that the user is forced to rely on special and expensive hardware to perform this kind of analysis.

In this paper we introduce kASA (*k*-mer Analysis of Sequences based on Amino acids), a fast, accurate and deterministic *k*-mer based tool for the analysis of metagenomic sequencing data with a very small, customisable memory footprint. The low storage requirement is achieved by an index that does not need to reside completely in the primary memory (RAM) but can remain largely on secondary memory like hard disk or solid state disk (using data structures from (14)). This feature makes kASA the first software tool capable of analysing complete metagenomic datasets on a standard commercially available notebook without losing accuracy or performance. kASA is written in `C++`, free, open source and available for all mainstream operating systems (Linux, Mac, Windows) and platforms (Desktop, HPCC, Notebook).

Unlike most other *k*-mer based tools, kASA constructs *k*-mers on an amino acid level by converting triplets of nucleotides via a given translation table that corresponds to the genetic code. This way, storing the *k*-mers requires less space (five bits instead of six), and in addition the sensitivity improves as synonymous DNA mutations (i.e. those that do not affect the encoded amino acid) no longer affect the matching process. In addition, we convert all genomic data (and not

DRAFT

only coding regions) to also allow not yet annotated sequence contigs as reference.
Another feature of kASA is the use of an entire range of word lengths k. Most other k-mer based methods use only a single k and thus have to make a singular decision how to the balance between sensitivity (small k) and precision (large k). With kASA the user can specify a minimum and a maximum k and the tool will try to match all word lengths of this closed interval. By dynamically adapting the word length k kASA is able to optimise both sensitivity and precision. If a k-mer is too short for being specific for a particular organism it may become specific with longer word length. On the other hand, if a longer k-mer cannot be found in the index due to a mutation, it could possibly be still found when using a shorter word length. The index is designed in a way that its size does not increase when using multiple word lengths.
These capabilities allow kASA to compete with or even outperform other tools.

Results and discussion

Pre-Processing

The comparison of NGS data with a database can be significantly accelerated by pre-processing the database into an index data structure. The index building step in kASA is depicted in Figure 1.

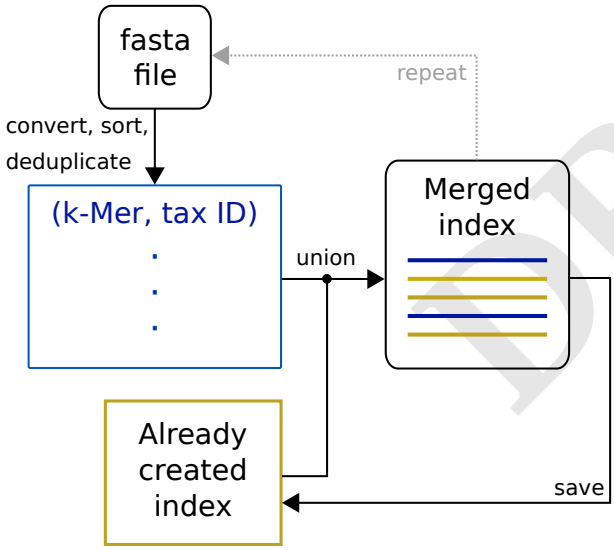

Fig. 1. Flowchart for creating the index. In the first step, as much DNA from a fasta file as possible is transformed into a container of k-mers which is then stored onto the hard drive as a pre-processed version of the index. If there is any DNA left in the file, this step is repeated and the resulting container is merged with the existing one on the drive. The process iterates until there is no data left in the file.

The DNA sequences from the database are scanned in all three reading frames and converted via a given table to amino acid sequences, including coding and non-coding regions. From the translated sequences all overlapping 12-mers (i.e. $k = 12$ amino acids, the maximum k supported by kASA) are extracted and saved into a file stored in secondary memory, together with the taxonomic ID of their source. After sorting and merging redundant entries (those, who share the same 12-mers and taxonomic ID), the index is ready to be used for identifying or profiling NGS data. Note, that this implies having multiple taxa for the same k-mer is allowed. In order to avoid having to rebuild the index every time the database is changed, kASA supports the addition of new reference sequences to the existing index without rebuilding.
In short the processing of sequencing reads works as follows: First the reads and their reverse complements are converted in the same manner as the database sequences. Then a set-intersection-like algorithm is used to find matches between k-mers from reads and the index, starting with the smallest k. If a hit is found, k is increased until the k-mers stop matching or the maximum k is reached. Any match is marked and scored. A more detailed description of these algorithms can be found in the Methods section.

Taxonomic identification of simulated reads

To evaluate the robustness of each tool against sequencing errors, we simulated sequencing data by sampling reads from 2500 randomly chosen bacterial genomes of known taxa with ART (15) (1026811 reads) and mason (16) (80211 reads). Since these genomes also serve as a reference, this test can be considered a sanity check. We also generated a negative control with reads of two species that were not part of the database used (Arabidopsis thaliana and Phaeodactylum tricornutum). A detailed description of our evaluation methods, the tools used and their settings, thresholds and formulas can be found in the Materials section and in supplementary files A and B.

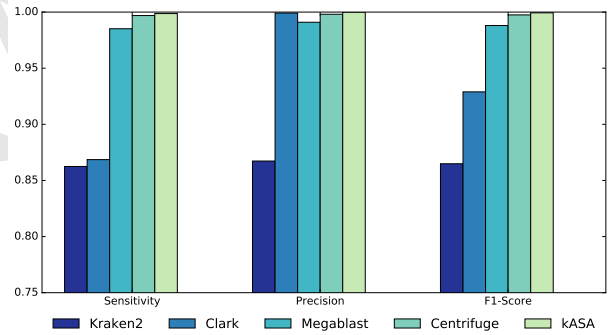

Fig. 2. Accuracy measurement of simulated data on species level. The shown measures range from 0.75 to 1 to stress differences and were averaged from the results of both tests. Specificity can be seen in the supplementary file A. Tools are sorted by F1 score.

Figure 2 and 3 show, that all tools are sufficiently capable of recognising slightly changed contents of their database. kASA has on average the highest F1-Score on the species level and genus levels followed by Centrifuge and MegaBLAST. The increase in sensitivity and precision on genus level for all tools except Clark is due to the simulated mutations and sequencing errors often making closely related species a better match. This also explains why exact matching algorithms operating on DNA with a fixed k are significantly worse than algorithms that support sequence alignments with gaps like Centrifuge or MegaBLAST. The significant increase in sensitivity and precision of Kraken2 stems from its method of abstracting to genus level if it cannot pick a certain species.

DRAFT

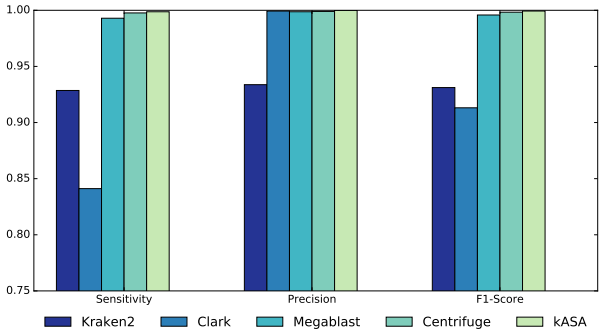

Fig. 3. Accuracy measurement of simulated data on genus level. For consistency, the same order, value range and colour as in Figure 2 is used. All tools except Clark increase their sensitivity and precision.

Our tool achieves superior accuracy even without explicitly modelling gaps since it is a k-mer based approach with dynamically adjusted word lengths. This ensures, that for a small enough k the non-mutated part matches. On top of that, we abstract to the amino acid level which makes at least one frame not likely to be affected by a single mutation due to robustness against synonymous mutations.

	Kraken Genus	**Clark** Genus	**kASA** Genus	**kASA** Species
HiSeq				
Sensitivity	0.82	0.81	0.98	0.98
Precision	0.97	0.98	0.99	0.99
F1 Score	0.89	0.89	0.98	0.98
MiSeq				
Sensitivity	0.80	0.80	0.99	0.94
Precision	0.91	0.91	0.99	0.94
F1 Score	0.85	0.85	0.99	0.94
simBA5				
Sensitivity	0.94	0.94	0.99	0.98
Precision	0.98	0.99	0.99	0.98
F1 Score	0.96	0.96	0.99	0.98

Table 1. Results for the mock community datasets (7). Results from Kraken and Clark with word length $k = 23$ are displayed here.

This advantage becomes especially visible when we apply kASA to the reference datasets used in (5, 7). The data sets "HiSeq" and "MiSeq" are described there as metagenomes containing sequences from ten genomes with the same abundance and the data set "simBA5" as consisting of bacterial and archaeal sequences sampled from replicons of 607 genera with mason.

A comparison between Table 1 and the results presented in the publication of Clark (7) shows that kASA has often both better sensitivity and precision than Kraken and Clark even if it operates on a lower taxonomic level (species rather than genus).

Because kASA was designed with minimal memory requirements in mind, we examined the memory consumption of each tool during index building and identification, as well as its index size. Results show (see Figure 4), that only kASA and MegaBLAST would be able to build up an index on a

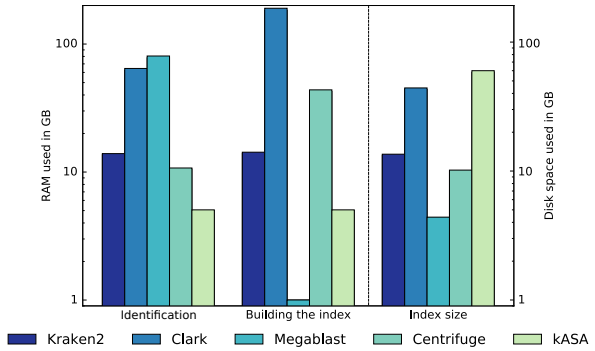

Fig. 4. Memory and space, which is used to build the index or identify the test data, and sizes of indices. The amount of RAM used by every tool as well as the amount of disk space in GB used to store the index is shown in logarithmic scale. Colours and order are equivalent to the ones in Figure 2 and Figure 3. Memory and space requirements of Kraken2 are directly correlated. kASA is shown with its minimum requirement of 5 GB. Without RAM restrictions, kASA uses 51 GB for identification and profiling, and as much as is available for index building which was 99 GB in our case.

standard notebook with 8GB RAM. Even if pre-built indices were used, no other tool than kASA would be able to identify any data on this platform.

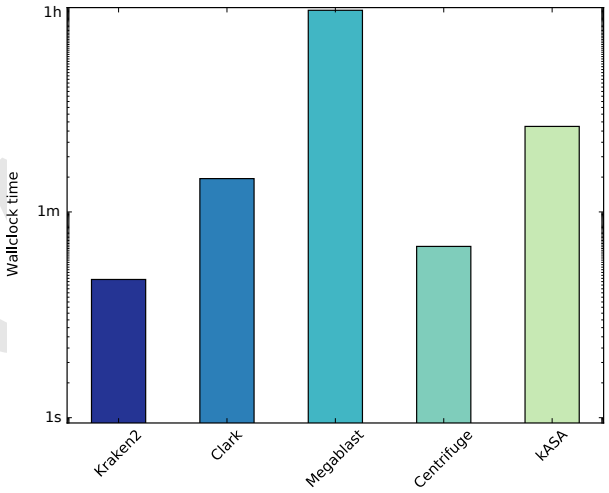

Fig. 5. Wall clock time for the accuracy tests of kASA and other tools. Times were averaged for both tests and sorted in the same manner as in Figure 2 and 3. The plot has a logarithmic y-Axis marking seconds, minutes, and hours. kASAs index was copied into RAM beforehand, which took about 83s. Omitting read identification and creating only a profile speeds kASA up by a factor of ∼1.3. Detailed measurements e.g. for performance on a Notebook can be found in supplementary file C.

Regarding computational speed, Figure 5 shows that Kraken2 is on average the fastest tool and MegaBLAST the slowest. kASA ranks midfield in the speed comparison; it ran on the same platform as the other tools and took less than six minutes on average. Reducing the memory limit to 5GB increases the run time by factor ∼1.6 in most scenarios. Regardless, the run times remain feasible.

Taxonomic profiling of simulated data

A taxonomic profile is a list of all organisms that have been identified in a dataset together with their respective abundance. This abundance is estimated by normalising the relative frequency against genome length and ploidy.

Consistent with other studies (e.g. (17)), we use the following method to calculate the relative frequency of each taxon: Given an identification of the reads, we determine the most significant taxon for each read and count the number of reads per taxon. We then divide these read counts by the total number of reads in the sample. This relative frequency is sufficient to compare the accuracy between tools, so we do not normalise against the genome lengths and ploidy.

Apart from this method, kASA also offers a direct procedure for creating profiles by counting k-mers that are specific for each taxon and dividing the k-mer counts by the total number of uniquely matched k-mers in the sample. If a k-mer is not specific, each mapped taxon gets an equal proportion to their count which is then divided by the total number of non-uniquely mapped k-mers in the sample. Depending on the ambiguity of the index, either of the relative frequencies may be relevant.

If taxonomic profiling is the main goal of a study, it is possible with kASA to completely dispense of the identification of the reads since this speeds up processing by a factor of $\sim$1.3. Furthermore, we advise to shrink the index by discarding a fixed proportion of k-mers from all taxa in equal proportions. This further speeds up the data processing without making significant sacrifices in terms of accuracy. For example, a reduction of the index by 50% decreases the F1-Score in our experiments to only 0.98 but increases performance by a factor of $\sim$1.4. A detailed investigation of the effect of different index sizes on the profiles can be found in supplementary file F.

Regarding the quality of taxonomic profiles, we evaluated kASA with the same data, methods and quality measures as used in a metastudy by Lindgreen et. al. (17). Although we did not run any of the tools studied in the paper except Clark again, we recalculated all measurements to ensure consistency (for more details, see supplementary file D).

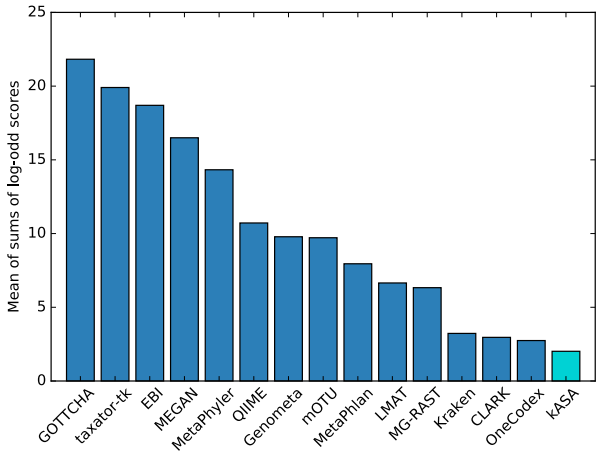

Fig. 6. Profile quality measured by the mean of log-odd scores. The log-odds of absolute differences between relative frequency and gold standard were summed per dataset and averaged regarding all six datasets. Tools are sorted by this score meaning lower is better. Data from the corresponding publication (17) was also used in the publication of Centrifuge (8) where the authors wrote, that their accuracy was similar to that of Kraken. The newest version of Clark was used to verify its own results as can be seen in supplementary file D.

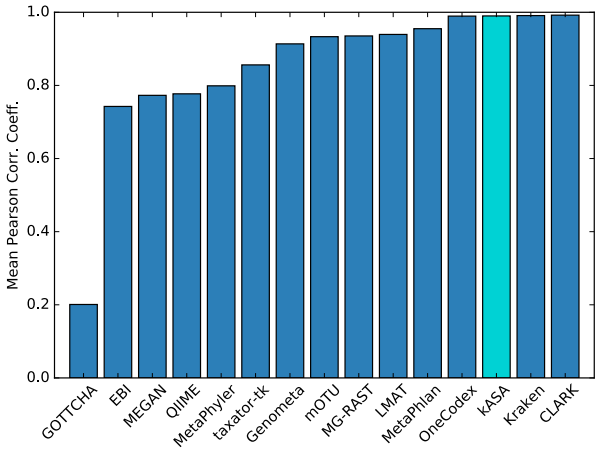

Fig. 7. Profile quality measured by the mean of Pearson Correlation Coefficients. The Pearson Correlation Coefficients of vectors of relative frequencies in relation to the respective truth for every dataset were averaged and shown here. Tools are sorted by that value. Results for the best four tools differ only slightly.

Figure 6 and 7 show that kASA is the best tool regarding log-odd scores and only slightly differs in value for the Pearson correlation coefficient. Relative frequencies for this comparison were gathered per read for consistency, as mentioned above, but profiles generated by a direct use of k-mer frequencies deviate only slightly (see supplementary file D sheet 2).

The CAMI-Challenge (18) proposes another comparison principle. It measures accuracy not only by difference from a gold standard but also by correctly identified taxa at every level. Combined with OPAL (19) it creates a framework for profile testing which offers the opportunity to compare kASA with even more tools. OPAL treats additional entries in the taxonomic profile as false positives even if their percentages are very low. To address this, we set a threshold of 0.001% in the profile on the species level.

Tool	Sum of scores
Quikr	36
kASA	39
MetaPhyler	48
FOCUS	60
MetaPhlan 2.0	62
TIPP	62
CLARK	64
mOTU	77

Table 2. Sum of scores given by OPAL for the CAMI dataset.

As shown in Table 2 and supplemental file E, kASA ranks second according to OPAL when using default settings. This indicates, that kASA would perform well enough for a CAMI-Challenge given the opportunity to discard low scoring entries.

Analysis of real data

To test kASA's ability to identify known organisms in real data, we used data from the Human Microbiome Project (20) to create taxonomic profiles and compare the results with

DRAFT

those created by Kraken2 (5, 6) equipped with the same database of size 105 GB. We chose Kraken2 because it was the only other tool able to build an index that large on our system and finish in a reasonable time.

Scientific Name	kASA u.	kASA n.-u.	Kraken2
Prevotella	43.5	27.9	30.7
Streptococcus	9.9	16.5	13.5
Neisseria	4.8	15.2	11.5
Veillonella	7.8	7.7	7.7
Porphyromonas	9.2	6.6	6.8
Haemophilus	3.9	5.1	4.5
Campylobacter	2.8	1.6	1.4

Table 3. Selection of genera with the highest relative frequency (in %) found in data sampled from human saliva (SRS147126). Results for kASA are shown with unique and non-unique values for k=12.

Table 3 shows that both tools agree in the most abundant genera although in different proportions depending on unique or non-unique counting. All detected genera are known to occur in the human oral flora which verifies our results. On phylum level both tools predict very similar proportions as shown in supplemental material G.

kASA can process reads of arbitrary length, so we were able to also analyse data from third generation sequencing technologies, namely from the Nanopore-Whole-Genome-Sequencing-Consortium (21) who used the Oxford Nanopore MinION technology (22) to sequence data from the GM12878 human cell line. Three samples were downloaded and analysed. kASA detected, as expected, the human genome as well as the Epstein-Barr virus. This leads to the conclusion that no contaminants were sampled which could have influenced an assembly. The taxonomic profiles are given in the supplemental material I. Note that all other taxa appearing in the profiles only have very low unique k-mer counts and are therefore likely artefacts due to sequencing errors or short similarities.

Conclusions

Our tests show that kASA performs taxonomic analyses at different taxonomic levels with excellent sensitivity and precision. The use of a dynamic k instead of a fixed word length as used in Kraken2 or Clark proves to be a valid method for improving sensitivity without sacrificing precision. The abstraction to amino acids further increases the robustness of kASA to mutations without the need for sequence alignment with gaps, such as in Centrifuge or MegaBLAST. This shows that exact matching algorithms can compete or even exceed BWT-based approaches.

In addition, kASA is, to our knowledge, the only available tool providing a RAM consumption so highly adjustable that it also works on cheaper computers at reasonable speed and without loss of accuracy. It therefore allows scientists who do not have access to expensive hardware to run their analysis without much effort or help from other sources. Since our approach relies heavily on secondary memory, using a solid state disk is the best way to increase throughput.

kASA can be used for taxonomic profiling, read analysis, and identification of contaminants. Although kASA would not be able to detect a completely new organism, it can take advantage of similarities between genomic sequences. Therefore we assume that the software would hint at organisms not present in the database as long as a sufficiently close relative is available.

Methods

Index creation

As mentioned before, pre-processing a database to create an index ensures that only necessary calculations are performed while identifying or profiling NGS data. This step (named 'build' in kASA) converts a nucleotide database in fasta-format to a binary file containing k-mers and saves it to secondary memory, e.g. the hard drive. Accessing this file during a comparison would create a lot of slow I/O operations, even when using a SSD. To keep the amount of these operations minimal, we build a prefix trie(23) from the index containing the first six coded letters of each k-mer. Each leaf of this trie contains two integers representing the upper and lower boundaries of the range containing k-mers that share the same prefix. If this prefix is matched, only a fraction of the index must be searched for the remaining suffix. This significantly reduces the search space and since the trie is small enough to fit into primary memory, prefix matches are performed very quickly.

Another important step in the construction of the index is the linking of DNA and its taxonomic ID. In order to recognise the origin of the DNA forming the nucleotide database, we use accession numbers from the fasta files and the NCBI taxonomy database. If there are no accession numbers or taxonomic IDs available, a dummy ID is given and the user receives a notification. The resulting so called "content file" is human readable and can be manually modified if necessary.

Identification and Profiling

After building the index, a comparison with NGS data can be performed. The algorithm is as follows:

First, the DNA and its reverse complement, or already converted amino acid sequences, from a fastq or fasta file are (translated and) converted into k-mers in a similar way as in 'build'. The differences here, are that each k-mer receives a read ID instead of a taxonomic ID and that duplicates are kept because they might hint at important motifs. For this reason, we advise the user and reader to de-duplicate reads beforehand to not distort abundances. The pairs are then sorted and passed to Algorithm 1 to see which taxa match to which read. An overview of this algorithm can be seen in Figure 8.

Note, that paired-end information will not be considered and all files will be treated as single-end. If kASA is given a RAM limit then some input files may be too large to be processed at once so the files are read in chunks.

The scores in algorithm 1 are calculated as follows: Every read which shares at least one k-mer with the index will be noted to create a relation from read ID to taxonomic ID.

DRAFT

```
input  : Sorted container I with converted k-mers, the
         prefix Trie and the index
output: Scores and counts

for Every k-mer x in I do
    Get prefix from x;
    Get range in index from trie with prefix;
    Set k to smallest value;
    Perform binary search in range with
      substring(x,k);
    if Match inside range then
        for Every entry from index in range do
            Get suffix from entry and concatenate with
              prefix to b;
            for k ← smallest value to highest value do
                if substring(x,k) ==
                  substring(b,k) then
                    Get all matching taxonomic IDs := T
                      and read IDs := R;
                    if |T| == 1 then
                        Count unique match;
                    else
                        Count non-unique match;
                    end
                    Calculate scores for T and R;
                else
                    Break;
                end
            end
        end
    end
end
return Scores and counts
```

Algorithm 1: Customised set intersection algorithm used for a comparison of NGS data with the index. It computes both the profile and the identification file per read. If the user is only interested in a taxonomic profile, a modified version of the depicted algorithm is used. It runs faster because it needs no bookkeeping of read IDs and thus allows for a better use of the available memory.

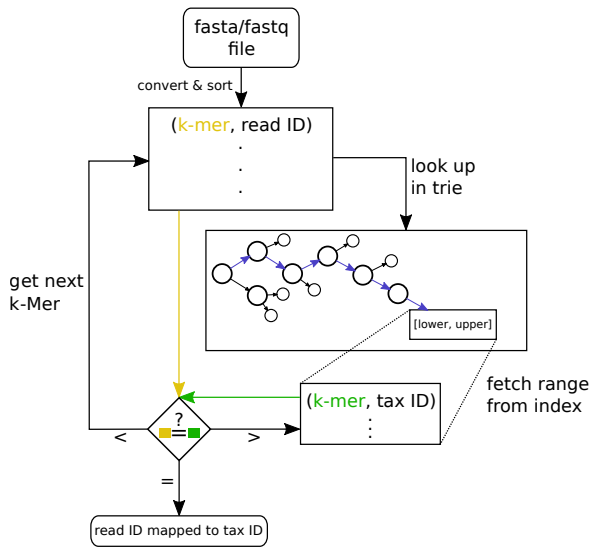

Fig. 8. Flowchart of the identification algorithm. The DNA and its reverse complement are converted into pairs of *k*-mer and read ID. After sorting, the prefix of the first *k*-mer determines the suffix range over the precalculated index. From this range, the first *k*-mer is compared with the one from the input. If they match, the read ID and the taxonomic ID are scored accordingly. If not, either the next *k*-mer from the input or the next index' *k*-mer is used. This loop continues until all *k*-mers from the input have been processed.

Since matches of smaller lengths are less significant than longer ones, this relation is represented with a weighted sum. Weights w_k are gained by a normalised quadratic function to stress a non-linear increase of significance. This means, that the values of k^2 are normalised to $(0,1]$ for $k = 1,...,12$. The sum using those *k*-dependent weights is called: *k*-mer Score of the taxon $t \in T$ and read $r \in R$

$$k\text{-mer Score}_{t,r} := \sum_{x \in I} \sum_{k=k_{small}}^{k_{large}} w_k \cdot \frac{|\text{matches with } \mathrm{x}_k(t,r)|}{1+\ln(|T|)}$$

where $|\text{matches with } \mathrm{x}_k(t,r)|$ is the number of *k*-mers that t and r share, and T and R are defined as in Algorithm 1. We used this weight function to strike a balance between rewarding long matches but not devaluing short matches too much.

Additionally, a second score called 'Relative Score' is derived from the *k*-mer Score in relation to the total number of *k*-mers of a taxon t present in the index (also called frequency of t) and the length of the read r:

$$\text{Relative Score}_{t,r} := \frac{k\text{-mer Score}_{t,r}}{1+\log_2(\text{length}(r) \cdot \text{frequency}(t))}$$

This formula is inspired by the calculation of the E-value in BLAST although in this case, a higher score indicates a better hit. The Relative Score can be used to determine the significance of a matched taxon. From our experience, for a read of length 100 everything with a Relative Score smaller than 0.5 can be seen as insignificant and for example sorted out during decontamination. For the output, the resulting scores are sorted in decreasing order by this relative score so that a read can have multiple identified taxa but the leftmost

DRAFT

one has the highest value.

The taxonomic profile that kASA computes consists of the names, taxonomic IDs and relative frequencies as well as the number of matched k-mers of all taxa found in a dataset. The number and relative frequencies are printed for each k and for unique and non-unique matches. This is necessary since the database used to form the index can be redundant for several reasons: For example an entry is named differently although it is identical to an existing one, or a subspecies is identical to another one on the amino acid level. kASA also offers a value describing the degree of redundancy or ambiguity present in the index which helps deciding which frequency to use. Note however, that for comprehensive databases this value may be overshadowed by large genomes.

Unique relative frequencies are calculated by dividing the number of uniquely matched k-mers for a taxon t by the total number of uniquely matched k-mers:

$$h_U\left(\text{Taxon } t\right)_k := \frac{\text{unique count}(t)}{|\text{uniquely matched k-mers}|}$$

Note, that "unique" is a local characteristic here since the input data may be processed in chunks. Global (as in for the whole data set) uniqueness can only be assumed if all converted k-mers fit entirely into the primary memory. However, judging from our experiments, this did not seem to influence the profiling quality significantly (e.g. Spearman's correlation coefficient for dataset A1 from (17) is $\rho = 0.96$).

If a k-mer x matched multiple taxa, the hit count (how often that k-mer matched) is divided by the number of matched taxa. The sum of these floating point numbers is divided by the total amount of matched k-mers at the end. The formula is as follows:

$$h\left(\text{Taxon } t\right)_k := \frac{\sum_{x \in I} \frac{\text{hit count}(t)}{|T|}}{|\text{matched k-mers}|}$$

If for example x was found five times in three different taxa, each of these three would receive $\frac{5}{3}$ to its dividend.

In conclusion, the identification file offers an answer to the question which organisms were found for each read, creating a basis for further studies. The taxonomic profile provides a broad overview of which organisms are present in the NGS dataset and how much DNA was contributed by them.

Materials

The database used for the experiments in (17) contained species which were no longer inside our version of the NCBI taxonomy and nt database. We manually fetched these entries via the E-utilities from the NCBI (24) and created a custom 'content file' containing the respective links to the taxonomic IDs. It can be found together with the results in supplementary file D. This procedure was used for experiments in table 1 as well.

Genomes used as database came from the NCBI nt database as of 2018-06-29 and contained all entries marked as bacteria, virus or archaea. Genomic data from the hg38 build (25) was added as well to include a large eukaryote. Since some tools could not build an index from a database that large, 2500 of 25687 taxa were selected randomly for our simulation. We ensured that all sampled species had a genus level so no misclassification could occur. Synthetic data was created from this smaller database with ART (15) and mason (16) and tests were conducted on an HPCC for a comparative basis. The full database (441 GB in size) and index (1.2 TB in size for kASA) were used for profiling real data, analysing files for Table 1 and 2, and creating Figure 6 and 7. To ensure maximum performance of kASA, the indices were shrunken via the lossless method described in the README file which can be found in our GitHub repository.

Abbreviations

RAM - Random Access Memory, **LCA** - least-common-ancestor, **SSD** - Solid State Drive, **I/O** - In- and Output, **BWT** - Burrows-Wheeler transform, **NGS** - Next Generation Sequencing, **HPCC** - High Performance Computing Cluster

Declarations

ACKNOWLEDGEMENTS

We thank Jan Grau and Matthew T. Weirauch for valuable conversations and Yvonne Poeschl for recognising problems where we least expected them. We would also like to thank the German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig for their support, especially in the initial phase of the project.

Funding

We acknowledge the financial support of the Open Access Publication Fund of the Martin Luther University Halle-Wittenberg.

Availability of data and materials

kASA is freely available on https://github.com/SilvioWeging/kASA and licensed under Boost Version 1.0. The content files and our artificial sequencing data created for our tests can be found here: https://doi.org/10.6084/m9.figshare.9015440

Author's contributions

SW and AGD designed the algorithms, SW implemented and tested them. Experimental design and test data creation was performed by SW and IG. SW, AGD, and IG wrote the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Ethics approval and consent to participate

Not applicable.

Consent to publish

Not applicable.

DRAFT

Bibliography

1. Philip Hugenholtz and Gene W. Tyson. Metagenomics. Nature, 455:481 EP –, Sep 2008. doi: 10.1038/455481a.
2. T. Laver, J. Harrison, P.A. O'Neill, K. Moore, A. Farbos, K. Paszkiewicz, and D.J. Studholme. Assessing the performance of the oxford nanopore technologies minion. Biomolecular Detection and Quantification, 3:1 – 8, 2015. ISSN 2214-7535. doi: https://doi.org/10.1016/j.bdq.2015.02.001.
3. Florian P. Breitwieser, Jennifer Lu, and Steven L. Salzberg. A review of methods and databases for metagenomic classification and assembly. Briefings in Bioinformatics, 09 2017. ISSN 1477-4054. doi: 10.1093/bib/bbx120.
4. Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A greedy algorithm for aligning dna sequences. Journal of Computational Biology, 7(1-2):203–214, 2000. doi: 10.1089/10665270050081478. PMID: 10890397.
5. Derrick E. Wood and Steven L. Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biology, 15(3):R46, Mar 2014. ISSN 1474-760X. doi: 10.1186/gb-2014-15-3-r46.
6. Derrick E. Wood and Steven L. Salzberg. Kraken2.
7. Rachid Ounit, Steve Wanamaker, Timothy J. Close, and Stefano Lonardi. Clark: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. BMC Genomics, 16(1):236, Mar 2015. ISSN 1471-2164. doi: 10.1186/s12864-015-1419-2.
8. Daehwan Kim, Li Song, Florian P. Breitwieser, and Steven L. Salzberg. Centrifuge: rapid and sensitive classification of metagenomic sequences. Genome Research, 26(12):1721–1729, 2016. doi: 10.1101/gr.210641.116.
9. Brian D. Ondov, Todd J. Treangen, Páll Melsted, Adam B. Mallonee, Nicholas H. Bergman, Sergey Koren, and Adam M. Phillippy. Mash: fast genome and metagenome distance estimation using minhash. Genome Biology, 17(1):132, Jun 2016. ISSN 1474-760X. doi: 10.1186/s13059-016-0997-x.
10. Brown et al. sourmash: a library for minhash sketching of dna. Journal of Open Source Software, 1(5):27, 2016. doi: doi:10.21105/joss.00027.
11. André Müller, Christian Hundt, Andreas Hildebrandt, Thomas Hankeln, and Bertil Schmidt. Metacache: context-aware classification of metagenomic reads using minhashing. Bioinformatics, 33(23):3740–3748, 2017. doi: 10.1093/bioinformatics/btx520.
12. Peter Menzel, Kim Lee Ng, and Anders Krogh. Fast and sensitive taxonomic classification for metagenomics with kaiju. Nature Communications, 7:11257 EP –, Apr 2016. doi: 10.1038/ncomms11257. Article.
13. Bethesda(MD). Nucleotide [internet]. National Library of Medicine (US), National Center for Biotechnology Information, 2004-.
14. Roman Dementiev, Lutz Kettner, and Peter Sanders. Stxxl: standard template library for xxl data sets. Softw., Pract. Exper., 38:589–637, 2008.
15. Weichun Huang, Leping Li, Jason R. Myers, and Gabor T. Marth. Art: a next-generation sequencing read simulator. Bioinformatics, 28(4):593–594, 2012. doi: 10.1093/bioinformatics/btr708.
16. Holtgrewe M. Mason - a read simulator for second generationsequencing data. Technical Report TB-B-10-06, Institut für Mathematik und Informatik, Freie Universität Berlin, 06 2010.
17. Stinus Lindgreen, Karen L. Adair, and Paul P. Gardner. An evaluation of the accuracy and speed of metagenome analysis tools. Scientific Reports, 6:19233 EP –, Jan 2016. Article.
18. Alexander Sczyrba, Peter Hofmann, Peter Belmann, David Koslicki, Stefan Janssen, Johannes Dröge, Ivan Gregor, Stephan Majda, Jessika Fiedler, Eik Dahms, Andreas Bremges, Adrian Fritz, Ruben Garrido-Oter, Tue Sparholt Jørgensen, Nicole Shapiro, Philip D. Blood, Alexey Gurevich, Yang Bai, Dmitrij Turaev, Matthew Z. DeMaere, Rayan Chikhi, Niranjan Nagarajan, Christopher Quince, Fernando Meyer, Monika Balvociute, Lars Hestbjerg Hansen, Søren J. Sørensen, Burton K. H. Chia, Bertrand Denis, Jeff L. Froula, Zhong Wang, Robert Egan, Dongwan Don Kang, Jeffrey J. Cook, Charles Deltel, Michael Beckstette, Claire Lemaitre, Pierre Peterlongo, Guillaume Rizk, Dominique Lavenier, Yu-Wei Wu, Steven W. Singer, Chirag Jain, Marc Strous, Heiner Klingenberg, Peter Meinicke, Michael D. Barton, Thomas Lingner, Hsin-Hung Lin, Yu-Chieh Liao, Genivaldo Gueiros Z. Silva, Daniel A. Cuevas, Robert A. Edwards, Surya Saha, Vitor C. Piro, Bernhard Y. Renard, Mihai Pop, Hans-Peter Klenk, Markus Göker, Nikos C. Kyrpides, Tanja Woyke, Julia A. Vorholt, Paul Schulze-Lefert, Edward M. Rubin, Aaron E. Darling, Thomas Rattei, and Alice C. McHardy. Critical assessment of metagenome interpretation–a benchmark of metagenomics software. Nature Methods, 14:1063 EP –, Oct 2017.
19. Fernando Meyer, Andreas Bremges, Peter Belmann, Stefan Janssen, Alice Carolyn McHardy, and David Koslicki. Assessing taxonomic metagenome profilers with opal. bioRxiv, 2018. doi: 10.1101/372680.
20. The Human Microbiome Project Consortium. A framework for human microbiome research. Nature, 486:215 EP –, Jun 2012. Article.
21. Miten Jain, Sergey Koren, Karen H. Miga, Josh Quick, Arthur C. Rand, Thomas A. Sasani, John R. Tyson, Andrew D. Beggs, Alexander T. Dilthey, Ian T. Fiddes, Sunir Malla, Hannah Marriott, Tom Nieto, Justin O'Grady, Hugh E. Olsen, Brent S. Pedersen, Arang Rhie, Hollian Richardson, Aaron R. Quinlan, Terrance P. Snutch, Louise Tee, Benedict Paten, Adam M. Phillippy, Jared T. Simpson, Nicholas J. Loman, and Matthew Loose. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nature Biotechnology, 36:338 EP –, Jan 2018.
22. Miten Jain, Hugh E. Olsen, Benedict Paten, and Mark Akeson. The oxford nanopore minion: delivery of nanopore sequencing to the genomics community. Genome Biology, 17(1):239, Nov 2016. ISSN 1474-760X. doi: 10.1186/s13059-016-1103-0.
23. Peter Brass. Advanced Data Structures. Cambridge University Press, 2008. doi: 10.1017/CBO9780511800191.
24. Bethesda(MD). Entrez Programming Utilities Help [Internet]. 2010-.
25. Human grch38.
26. Brian D. Ondov, Nicholas H. Bergman, and Adam M. Phillippy. Interactive metagenomic visualization in a web browser. BMC Bioinformatics, 12(1):385, Sep 2011. ISSN 1471-2105. doi: 10.1186/1471-2105-12-385.

Supplementary material

A. 1_FVP.pdf

Formulas for sensitivity, precision and specificity, versions and parameters of used tools as well as system specifications.

B. 2_SimResults.ods

Table of results for all simulated experiments for both species and genus level.

C. 3_TimesAndMemory.ods

Table with results from run time and memory consumption tests.

D. 4_QuantResults.ods

Table with results from the Lindgreen et. al. quantification test.

E. 5_OPAL.zip

HTML file created by OPAL to visualise the results for the first CAMI Challenge.

F. 6_Shrink.zip

Study of accuracy with shrunken indices.

G. 7_RealDataResults.ods

Results of Kraken2 and kASA from data sampled from human saliva.

H. 8_KronaRealData.html

HTML file created by Krona (26) for results of data from human saliva.

I. 9_ProfilesNanopore.zip

Profiles of data from the Nanopore-Whole-Genome-Sequencing-Consortium.

DRAFT