Cleanifier: A fast and lightweight k-mer based tool to remove contamination in microbial sequence data

Jens Zentgraf,[†,1,2,3] **Johanna Elena Schmitz**[†,1,2,3] **and Sven Rahmann**[1,3*]

[1]Algorithmic Bioinformatics, Saarland University, Campus E2.1, 66111, Saarland, Germany, [2]Saarbrücken Graduate School of Computer Science, Saarland University, Campus E2.1, 66111, Saarland, Germany and [3]Center for Bioinformatics Saar, Saarland University, Campus E2.1, 66111, Saarland, Germany

*Corresponding author. rahmann@cs.uni-saarland.de

†Both authors contributed equally and share first author ship.

FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Summary: When working with sequence data of human microbiomes, the first step in most analysis pipelines is the removal of human contamination. There are two reasons why the removal of human reads is a critical step in microbiome data analysis. First, many countries have strict privacy and data protection guidelines for human sequencing data. This means that microbiome data containing partly human data cannot be processed outside a hospital and cannot be published. The second and from a biological view more important reason is that human contamination may cause problems in later analysis steps, such as genome assembly and binning. Current tools are either alignment- or k-mer based. In alignment based methods, all reads are aligned against the human reference and classified as human contamination based on the alignment score of the reads. However, the alignment step needs a significant amount of time. In contrast, k-mer based tools are usually faster, because they classify the reads by querying all k-mers of the reads in an efficient reference k-mer database.

We introduce Cleanifier, a fast and lightweight alignment-free tool to detect and remove human contamination. For this, we create an pangenome index of all human gapped k-mers, also called spaced seeds, that is made available online via Zenodo. If users are interesting in microbiome data from other hosts, they can alternatively build their own index using other reference genomes. In the filtering step, we query all k-mers of a read in the index and classify it as human based on the fraction of read base pairs covered by a k-mer. The human reads are either removed or written into a separate file for later analysis.

Availability and Implementation: The implementation is available via gitlab (`https://gitlab.com/rahmannlab/cleanifier`), PyPi (`https://pypi.org/project/cleanifier/`) and Bioconda (`https://anaconda.org/bioconda/cleanifier`).

Contact: rahmann@cs.uni-saarland.de

Key words: alignment free, k-mer, spaced seeds, gapped k-mer, hashing, probabilistic filter, contamination removal, sequence data, microbiome, metagenomics

Introduction

The removal of host contamination is a critical first step in most metagenomics pipelines that work with human microbiomes due to both privacy concerns and to avoid complications in downstream analysis [1], see 1. For example, many public databases, such as the NCBI Sequence Read Archive (SRA) [2], require that human-derived sequence data is free of human reads prior to publishing. Given the ever-growing amount of microbiome data [3], an easy-to-use, fast, and lightweight tool to remove host contamination is of high interest. Current tools include both alignment based and alignment free k-mer based methods, see Table 1. Alignment-based tools use aligners such as Bowtie2 [4], Minimap2 [5] or BWA [6], to remove all reads that map to the human reference. For example, KneadData [7] uses Bowtie2 and Hostile [8] uses Bowtie2 for short reads and Minimap2 for long reads. To avoid the slow alignment step, k-mer based methods are a common alternative with a similar accuracy. The taxonomic classifiers Kraken 2 [9] and KrakenUniq [10] are most frequently used to first classify the reads and then filter out all reads that were classified as human. Another k-mer based tool is the Human Read Removal Tool (HRRT), which is based on the Sequence Taxonomic Analysis Tool (STAT) [11].

We here introduce Cleanifier, a fast and accurate k-mer based method that builds a gapped k-mer index containing all gapped k-mers from the human T2T genome [12], all 47 pangenome assemblies from the Human Pangenome Reference Consortium [13], common SNPs from the 1000 Genome Project [14] and all HLA variants stored in the IPD-IMGT/HLA database [15]. Since we provide the index online, users only need to perform the fast filtering step using a simple command-line tool supporting single-read, paired-end, short and long read sequencing data both in compressed and uncompressed FASTQ format.

© The Author 2025. Latex template adapted from Oxford University Press.

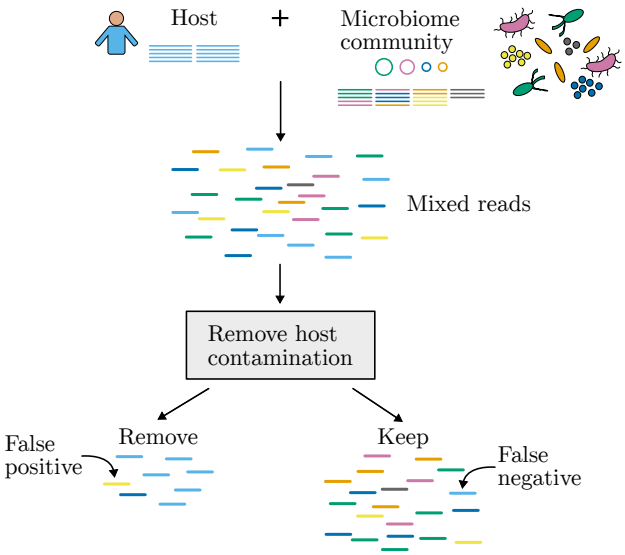

Fig. 1: Sequencing the microbiome from a human body site, such as the gut or skin, leads to human contamination in the sequencing dataset. The goal of Cleanifier is to remove all human reads from the sequencing sample prior to publishing or downstream analysis. Wrong classification of human reads can either lead to false positives or false negatives. False positives cause a reduced microbiome retention while false negatives lead to privacy issues if not all human reads are correctly removed.

In the next sections, we first describe some preliminaries and the indexing and filtering step of Cleanifier. We then compare Cleanifier with Hostile (alignment based), Kraken 2 (k-mer based taxonomic classifier), noHuman (Kraken 2 wrapper with custom database) and HRRT (k-mer based).

Preliminaries

Before we describe the method in more detail, we first introduce multiple basic definitions, such as k-mers and the used data structures. Considering the DNA alphabet $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$, a sequence $s \in \Sigma^n$ of length $|s| = n$ consists of the 4 DNA bases Adenine (`A`), Cytosine (`C`), Guanine (`G`) and Thymine (`T`).

Definition 1 (k-mer) A k-mer is a sequence of length k over a given alphabet Σ. For a sequence $s \in \Sigma^*$ with $|s| \geq k$, a substring of length k is called a k-mer in s. A k-mer can start at any position $< |s| - k$ in the sequence.

Since the DNA is double-stranded, the reverse complement of a DNA sequence, and respectively the reverse complement of a k-mer, corresponds to the same molecule and should thus be treated as equivalent.

Definition 2 (Reverse complement) Given a DNA sequence (or k-mer) s, its reverse complements $rc(s)$ is given by inverting the base order and replacing `A` with `T`, `C` with `G` and vice versa.

Since the DNA alphabet consist of 4 different bases, we need $log_2(4) = 2$ bits to encode them. Hence, to store a k-mer efficiently, we transform the k-mer to an integer using an canonical integer encoding that returns a unique value for each k-mer and its reverse complement.

Definition 3 (Canonical integer encoding of a k-mer) Each nucleotide is encoded by two bits ($\texttt{A} = (0)_4 = (00)_2$, $\texttt{C} = (1)_4 = (01)_2$, $\texttt{T} = (2)_4 = (10)_2$ and $\texttt{G} = (3)_4 = (11)_2$). This encoding $enc(s)$ gives us a unique base-4 number for each k-mer. To account for the reverse complement, we take the maximum of the base-4 number of the k-mer and its reverse complement as the canonical integer code $cc(s) = max(enc(s), enc(rc(s)))$.

```
TACGCGTAAG
k-mers        reverse complements     canonical code
TACGC          rc(TACGC) = GCGTA      cc(TACGC) = 888
 ACGCG         rc(ACGCG) = CGCGT      cc(ACGCG) = 478
  CGCGT        rc(CGCGT) = ACGCG      cc(CGCGT) = 478
   GCGTA       rc(GCGTA) = TACGC      cc(GCGTA) = 888
    CGTAA      rc(CGTAA) = TTACG      cc(CGTAA) = 647
     GTAAG     rc(GTAAG) = CTTAC      cc(GTAAG) = 899
```

A k-mer is a contiguous substring of length k. Hence, a single sequencing error in a read changes k consecutive k-mers. To be more tolerant against errors, previous alignment-free methods proposed to used so called gapped k-mers or spaced seeds that consider k significant positions in a window of size w [18, 9, 19].

Definition 4 (Gapped k-mer) Given a string of length w and a mask of shape (w, k) over the alphabet $\{\texttt{\#}, _\}$ with k times the character `#` and $w - k$ times the character _, a gapped k-mer is a string of length k where $w - k$ positions are masked (insignificant). The k significant positions are marked by `#` in the mask.

We only consider symmetric masks where the first and last position are significant to get the same canonical integer encoding for the forward and reverse strand. Since the robustness against errors depends on the positions of the significant and insignifcant position, we only consider masks with good error prooperties [20].

For example, given the string `TACGCGTAA` and the mask `##_#_##`, we can extract the following canonical gapped k-mer codes.

```
TACGCGTAAG
TA_G_GT     cc(TAGGT) = max(enc(TAGGT), enc(ACCTA)) = 88
 AC_C_TA    cc(ACCTA) = max(enc(ACCTA), enc(TAGGT)) = 88
  CG_G_AA   cc(CGGAA) = max(enc(CGGAA), enc(TTCCG)) = 496
   GC_C_AG  cc(TAGGT) = max(enc(GCCAG), enc(CTGGC)) = 445
```

Method

Cleanifier first builds an index data structure that contains all human (gapped) k-mers. Given an index and a (paired-end) sample, Cleanifier then classifies all reads in a FASTQ file and writes new FASTQ files that only contain the microbial reads. Subsequently, we describe first the index building step and then the filtering (classification) step. The index building step only needs to be performed once per host organism and we provide the described human index online.

Index data structures

Cleanifier supports both a multiway bucketed Cuckoo hash table and a windowed Cuckoo filter for creating the (gapped) k-mer index. Cuckoo hashing has already been used in prior work due

Tool	Ref.	Single-end	Paired-end	Short reads	Long reads	Compression	Alt. host
Hostile	[8]	✓	separate	✓	✓	✓	✗*
HRRT	[11]	✓	interleaved	✓	✓	✗	✗
noHuman	[16]	✓	separate	✓	✓	✓	✗
Kraken 2 + Krakentools	[9, 17]	✓	separate	✓	✓	input only	✓
Cleanifier	[this]	✓	separate	✓	✓	✓	✓

Table 1. Overview of human read removal tools and supported input data. Paired-end data is either required as two separate files or as interleaved paired-end format. The standard Kraken 2 database may be used to remove any host and Cleanifier can be used to remove other host organisms by building a custom database. The remaining tools don´t provide the option to build a custom database. *Hostile currently supports the removal of human or mouse reads.

to their low memory consumption, fast lookup times and support of online insertions, i.e., the whole key set is not required to be known in advance [21, 19].

Cuckoo hash table

In (d, l)-bucketed Cuckoo hashing, an array with n slots is divided into $b = \lceil n/l \rceil$ buckets with l slots each. To store a key $x \in \mathcal{U}$, we use $d \geq 2$ hash functions $f_1, \ldots, f_d : \mathcal{U} \to [b] := \{0, 1, \ldots, b-1\}$ to compute d possible bucket addresses where the key x may be stored. Hence, a key together with its optional value can be stored in any of the $d \cdot l$ slots. In the most common configuration of $(3, 4)$ Cuckoo hashing, a table with n slots can be filled up to $\approx 98\%$, leading to a low memory overhead [22].

When checking the presence of a key in the table, all $d \cdot l$ slots are searched, until either the key or a free slot is found. Since all slots of one bucket are usually contained in the same cache line, we only have up to d cache misses per lookup, with d usually set to 2 or 3.

The insertion of a key x works as follows: If we find an empty slot in one of the buckets given by $f_1(x), \ldots, f_d(x)$, the key is inserted in the first free slot. If no free slot is found, we remove a random key from one of the d buckets and try to re-insert it in one of its alternative buckets. This step is repeated until either a free slot is found or a maximum number of steps is reached. The maximum number of steps guarantees that insertions are done in a constant time. If an element cannot be inserted, we have to create a bigger hash table (if the table is full) or use different hash functions and redo the insertion for all elements.

Probabilistic Cuckoo filter

Probabilistic filters are space efficient data structures to answer membership queries fast for a set of keys S from a universe $\mathcal{U}$. If we query a key $x \in S$, the filter always correctly reports the key as present. However, if we query a key $x \notin S$, the filter may erroneously report the key as present with a small controllable error probability ε, called false positive rate (FPR). The FPR is closely linked to the space requirements of the filter, i.e., the required space increases for lower FPRs. Cuckoo filters use the same collision resolution strategy as Cuckoo hash tables and thus have the same advantages as Cuckoo hash tables, but with the additional benefit of requiring less space due to storing only a small p-bit fingerprint instead of an exact representation of the key x. Since we compare a p-bit fingerprint with all fingerprints stored in the $d \cdot l$ possible slots, the probability to falsely report a key as present is $2^{-p} \cdot d \cdot l$.

We here use $(2, 2)$ Cuckoo filters with a windowed memory layout, since they need less space compared to bucketed Cuckoo filters for the same FPR [23]. The (d, l) windowed Cuckoo filters work similar to a bucketed Cuckoo filter, but the n slots are divided into $n - l + 1$ overlapping windows instead of buckets. At each slot a new window starts and overlaps with the next $l - 1$ windows. It has been shown to increase the maximum fill rate of Cuckoo filters [22, 23], resulting in ≈ 17 bits per key for a FPR of $2^{-14} \approx 0.00006$.

Indexing the human genome

In the indexing step, all (gapped) k-mers of the host genome are inserted into the index. To be precise, the index stores the canonical integer representation of the (gapped) k-mers present in the provided reference sequences. To summarize, the index data structure can either be a bucketed Cuckoo hash table (stores exact (gapped) k-mer code) or a windowed Cuckoo filter (stores a p-bit fingerprint for each (gapped) k-mer; here configured to have a FPR of $2^{-14} \approx 0.00006$). The advantage of the Cuckoo filter is the lower space requirement compared to the Cuckoo hash table, at the cost of a small FPR, which we show to be negligible in practice (see Results).

Advanced Human Precomputed Index

We provide a precomputed index for filtering human reads. The index is based on the T2T-reference [12] which provides the benefit that it does not contain undefined bases. To increase the sensitivity against common variants, we include the variants from the 1000 Genome Project [14]. For this, we analyze the VCF (based on the T2T-reference) for all 3202 genomes, extract all variants with an allele frequency of at least 1% and add all k-mers which overlap these variants to the index. By using gapped k-mers, the approach is already more robust against SNPs which is further improved by including the variants. One problem of gapped k-mers is that they are robust against substitutions but not against insertions and deletions. By including all variants based on common INDELs, we try do compensate this problem. In addition, we include all 47 pangenome assemblies from the Human Pangenome Reference Consortium [13]. Since the MHC is a highly complex region in the genome with many variants, we further add all HLA variants stored in the IPD-IMGT/HLA database [15].

For a comparison between an index that only contains the gapped k-mers of the T2T reference and our enhanced index, see Supplementary Materials.

Filtering microbiome samples

To decide whether a read belongs to the host organism or a microbiome, we compute and query the (gapped) k-mers in the read s. Given a threshold $T \in [0, 1]$, a read is classified as human if more than $T \cdot |s|$ bases are covered by (gapped) k-mers found in the human index.

Cleanifier supports to classification modes, a fast sampling mode that only queries a subset of the (gapped) k-mers and a

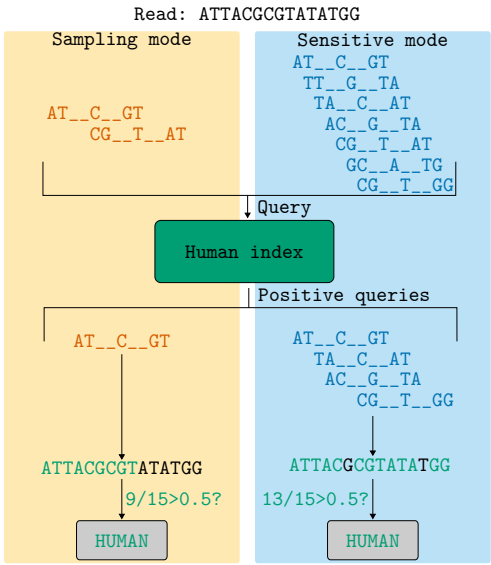

Fig. 2: Cleanifier classifies a read by querying either a subset (sampling mode) or all (sensitive mode) gapped k-mers of the read in a human index. If a gappend k-mer is present in the index, then the corresponding bases in the read are set to covered. In the sampling mode, all bases covered by the whole window count as covered. By contrast, the sensitive mode only considers the significant positions as covered. If the covered bases exceed a user-defined threshold (default 0.5), the read is assumed to be human contamination and filtered out.

sensitive (but slower) mode in which all (gapped) k-mers of the read are queried in the human index, see Figure 2.

Sampling mode

When we query a gapped k-mer of shape (w, k) in the index, we skip the next $\lfloor w/2 \rfloor$ consecutive gapped k-mers. Hence, every base has the chance to be covered by two distinct gapped k-mers. If the filter reports the queried gapped k-mer as present, we count all w positions as covered.

Sensitive mode

In the sensitive mode, we query all gapped k-mers in our index. If the filter finds the queried gapped k-mer, only the significant positions are marked as covered.

Paired-end Reads

To filter paired-end reads, each read of a pair is classified independently. If one of the two is classified as a human read, the whole pair is removed.

Implementation details

Cleanifier is implemented as command-line tool in Python using the `numba` [24] package to just-in-time compile the Python code. We use the Cuckoo filter implementation from [23] and the Cuckoo hash table from [19], which uses double quotienting to further reduce the space requirements of the hash table.

Parallelization

Our current implementation parallelizes both the indexing and filtering step of Cleanifier using a consumer-producer architecture. The threads communicate over lock-free buffers that use atomic volatile `ready_to_read` and `ready_to_write` flags.

Indexing step

To enable parallel insertion with t threads into our index without locking, we divide the Cuckoo hash table or filter, into t independent cache-aligned sub-arrays. An additional I/O thread reads the input sequences in buffers that are consumed by a producer threads that splits the read into its k-mer and writes them to its output buffers. The inserter threads then consune the output buffers of the splitter threads and perform the insertion of the k-mers into the index.

Filtering step

Similarly to the index building step, one I/O thread reads the (compressed) FASTQ files into multiple buffers. For paired-end data, each buffer is divided into two halves, the first half contains the reads of the first pair and the second half of the other pair. The classifier threads consume the buffers from the I/O thread, classify the reads by querying the gapped k-mers and write the results into their respective output buffers. A writer thread consumes all output buffers from the classifier threads and writes the reads that are not classified as human to the output FASTQ files.

Shared Memory

Cleanifier provides the option to load the index to shared memory. Therefore, the index is loaded to memory once and can be used by multiple independent Cleanifier instances. On the one hand, this saves the time to load the index to memory each time a Cleanifier process is started. On the other hand, this reduces the required memory and enables parallelization over multiple instances without an increasing memory footprint.

Results

First, we describe the human and microbiome data and the tools used in our evaluation. We compared all tools with respect to their filtering accuracy, running time and memory requirements on short and long read sequencing data.

Datasets

We evaluated the accuracy of the tools in terms of correctly filtering out human reads and retaining microbial reads. As we do not have a gold-standard human read removal tool, we evaluated the tools on human whole genome sequencing data, which should therefore only contain human reads, and on simulated microbiome data. The accession numbers are provided in the Supplementary Materials.

Human data

We downloaded whole human genome sequencing data for HG002 to HG006 cell lines provided by the Genome in a Bottle (GIAB) consortium [25]. We evaluated the performance for short reads on paired-end Illumina data, downsampled to contain 50 million reads each. For long reads, we downloaded Pacific Biosciences (PacBio) reads of HG002, downsampled to contain 5 million reads.

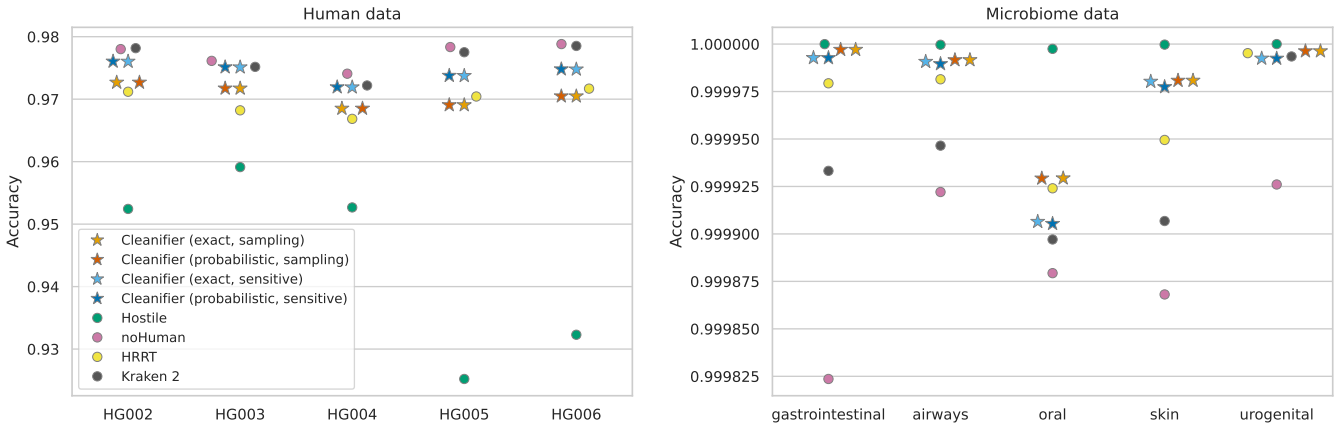

Fig. 3: Accuracy on Illumina paired-end short reads for human (correctly filtered out) and microbiome (correctly retained) data; higher is better.

Microbiome data

We downloaded the toy human microbiome data simulated for the CAMI 2 challenge [26, 27] for 5 different body sites (gastrointestinal, airways, oral, skin and urogenital) as a gold standard that contains only microbial reads. The data was simulated based on the microbiome profiles from the Human Microbiome Project [28]. We combined the available FASTQ files, such that each file per body site contains 50 million paired-end Illumina short reads and 5 million PacBio long reads, respectively.

Tools

We evaluated Cleanifier against four state-of-the-art methods for human read removal. We ran all tools using 8 threads and their respective default parameters, unless explicitly stated otherwise. We provide a brief description of the evaluated tools below.

- We evaluated Cleanifier using the index described above with a $(29, 33)$ gapped k-mer mask and a threshold of 0.5 for classification (default threshold, see Supplementary Materials). We ran four different Cleanifier versions: Cleanifier (exact, sampling), Cleanifier (probabilistic, sampling), Cleanifier (exact, sensitive) and Cleanifier (probabilistic, sensitive) for all combinations of the exact hash table and the probabilistic filter run in the sampling and sensitive classification mode.
- Hostile [8] is an alignment-based approach to remove human contamination by aligning all reads to the human genome using Bowtie2 for short reads and Minimap2 for long reads. We ran Hostile with the default index comprised of the human T2T reference [12] and HLA variants from the IPD-IMGT/HLA database [15].
- Kraken 2 [9] is a fast k-mer based taxonomic classification tool for metagenomics sequencing data. We evaluated Kraken 2 using the standard database (available for download at `https://benlangmead.github.io/aws-indexes/k2`). After taxonomic classification, reads that are not classified as *Homo sapiens* (taxonomy ID: 9606) are extracted using Krakentools [17].
- noHuman [16] is a Kraken 2 wrapper specifically build to remove human contamination. It uses a custom Kraken database built from all genomes available in the Human Pangenome Reference Consortium [13].
- HRRT is part of the SRA Taxonamy Analysis Tool (STAT) [11]. Reads are removed by querying k-mers in a MinHash-based index that contains all k-mers present in human derived eukaryotic species; excluding all k-mers that are known to also occur in non-eukaryotic species.

Settings

We ran all benchmarks on an Ubuntu (24.04 LTS) server with two AMD EPYC 9534 64-Core Processors, 1.5TB 4800 MHz DDR5 memory and a KIOXIA CD8P SSD. We measured running time and maximum memory usage with `/usr/bin/time -v`.

Accuracy on short reads

We compute the accuracy as the number of erroneously classified reads divided by the total number of reads, i.e, the incorrect retention of human reads and the erroneous removal of microbiome reads, for human and microbiome data, respectively. Figure 3 shows the accuracy for both human and microbiome data.

Kraken 2 and noHuman are very sensitive to human reads, resulting in the highest accuracy on human data, but the lowest accuracy on microbiome data. In contrast, Hostile is less sensitive to human reads, but has the best microbiome retention. All Cleanifier versions and HRRT have a high accuracy on both human and microbiome data, with Cleanifier having a slightly higher accuracy compared to HRRT.

Among the Cleanifier versions, the exact and probabilistic versions have almost the same accuracy on all samples, showing that for a threshold of 0.5, the small FPR incurred by the filter for a single k-mer query is negligible in practice, as several k-mer matches are required to classify a read as human. The sampling mode of Cleanifier is less sensitive to human reads compared to the sensitive mode. On average, 0.38% of all reads are additionally not correctly filtered out. On microbiome data, the sampling and sensitive mode have a comparable accuracy.

Since the microbial reads are simulated, they act as a gold standard. In contrast, the human data is not simulated and may therefore contain a small amount of contaminants, such as the Eppstein-Barr or PhiX virus. Figure 5 shows the overlap of "falsely" classified reads in the HG002 human dataset between Cleanifier and the other tools. All pairwise comparison can be

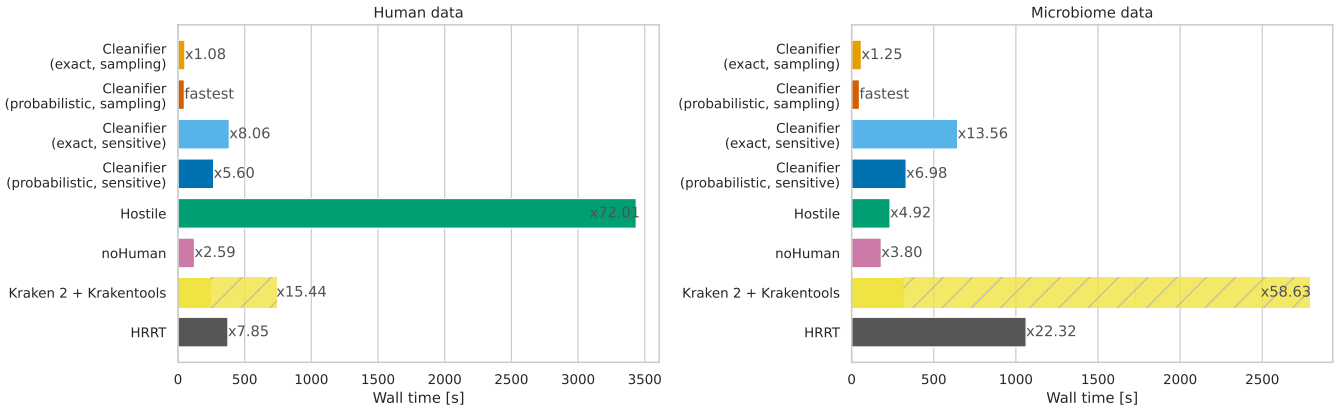

Fig. 4: Running time on short reads in seconds on datasets with 50 million paired-end reads. All tools are run with 8 threads. The running time is averaged over 5 human and microbiome samples, respectively.

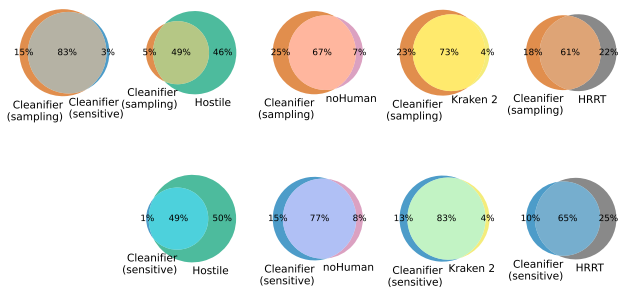

Fig. 5: Overlap between "falsely" retained human reads of HG002 between Cleanifier (probabilistic) and the other tools.

found in the Supplementary Materials. The reads classified as non-human by Cleanifier (both sampling, and sensitive mode) are almost a true subset of Hostile and share a high overlap with noHuman, Kraken and HRRT, suggesting that those reads may indeed be contaminants or poor quality reads.

Running time on short reads

We measured running time for reading and writing uncompressed FASTQ files and configured all tools to only output the microbial reads to prevent an unfair comparison due to I/O latency. Figure 4 shows the running time to filter FASTQ files with 50 million paired-end short reads using 8 threads, shown separately for human and microbiome reads. Speedup for using Cleanifier with different number of threads can be found in the Supplementary Materials. In practice, human contamination in microbiome samples can reach up to 90% [29], requiring tools that are fast for classifying both human and non-human reads.

On both human and microbiome data, Cleanifier (sampling) is fastest, while the sensitive mode is slower than both Cleanifier (sampling) and noHuman, which is the 2nd fastest tool. Taxonomic classification with Kraken 2 using the standard database it slower compared to the smaller database used in noHuman. The overall running time of Kraken 2 + Krakentools is significantly slower, due to the slow filtering with Krakentools. In particular, Krakentools is very slow if large FASTQ file have to be written as is the case for the microbiome data. In the case of human data, almost all data is discarded, resulting in a lower overhead of Krakentools. On human data, HRRT is 7.85 times slower than Cleanifier (sampling) and 22.32 slower on microbiome data. As expected, Hostile is slowest on human data (72 times slower than Cleanifier (sampling)), due to the slow alignment step, and faster on microbiome data since no full alignment is performed, if no start seed is found.

Accuracy on long reads

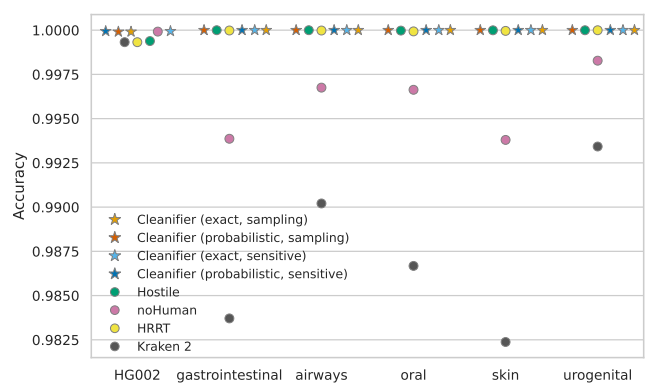

Fig. 6: Accuracy on PacBio long reads for human (correctly filtered out) and microbiome (correctly retained) data; higher is better.

Next, we measured the accuracy on long reads, see Figure 6. All tools, except Kraken 2 and noHuman, have an accuracy of ≈ 1.0 on microbiome data. While the accuracy on human short read data was < 0.98 for all tools, they have an accuracy $> 99\%$ on long reads, with all Cleanifier versions having an accuracy of ≈ 1.0, which is a modicum better compared to Hostile, HRRT and Kraken 2.

Running time on long reads

As expected, the relative running time between the tools is similar on long reads and short reads. Cleanifier (sampling) is fastest on

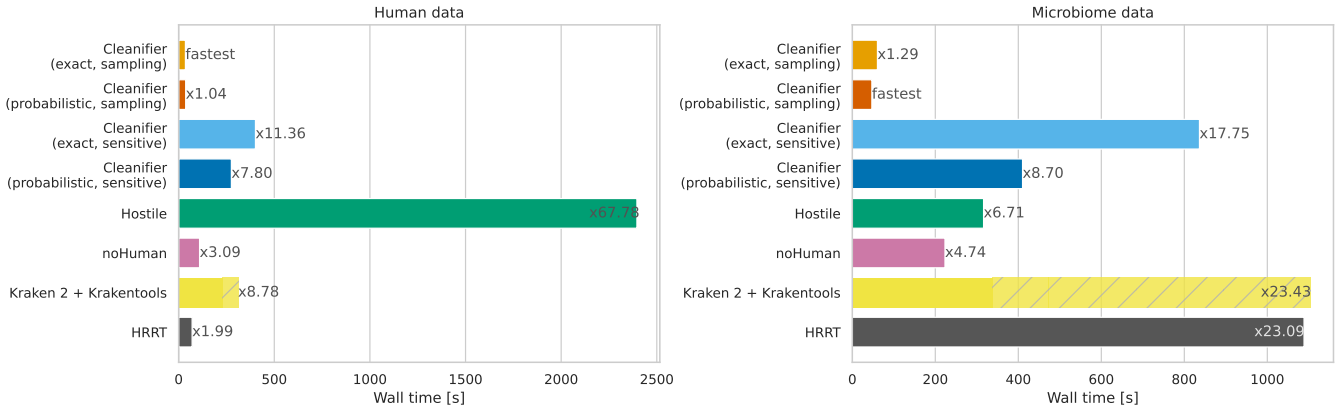

Fig. 7: Running time in seconds to filter 5 million long reads with 8 threads. The running time for microbiome data is averaged over the 5 samples.

both human and microbiome data, while Hostile is 67 times slower on human data. On microbiome data, HRRT and Kraken 2 + Krakentolls are slowest (by a factor of ≈ 23).

Memory requirements

	Cleanifier		Hostile	noHuman	HRRT	Kraken 2
	exact	prob.				
Human	13.85	6.90	3.86	5.22	4.37	88.83
Microbiome	13.85	6.90	3.65	5.38	1.00	90.46

Table 2. Maximum memory usage in GB; measured on short read samples.

Table 2 shows the memory requirements measured for filtering short read data. Hostile and HRRT have the smallest memory footprint. noHuman and Cleanifier (probabilistic) need a similar amount of memory of 5.3 GB and 6.9 GB, respectively. Cleanifier (exact) requires twice as much memory as Cleanifier (probabilistic). Kraken 2 using the standard database has the highest memory requirements of $\approx$ 90 GB. However, as seen in the case of noHuman, a smaller Kraken 2 database suffices to get a high accuracy. Thus, the memory usage can be greatly reduced by using a smaller Kraken 2 database.

Conclusion and discussion

We presented a new tool, called Cleanifier, to remove host contamination from microbiome sequencing experiments. Cleanifier is fast, memory-efficient and has a high accuracy. On both short and long read data, the sampling mode of Cleanifier makes it the fastest tool without loosing accuracy. In addition, Cleanifier (probabilistic) has low memory requirements < 7 GB independent of the input data.

The comparison of Cleanifier using a Cuckoo hash table (exact) and Cuckoo filter (probabilistic), shows that under the same circumstances, i.e. same references used to build the index and same classification method, the small FPR incurred by the probabilistic Cuckoo filter does not negatively effect the accuracy on both short and long reads. Although it should be noted that we only tested reads longer than 150 bp and an adverse effect may be visible on very short reads where a single positive k-mer query decides the classification of the read as human, in this case a worse microbiome retention may occur.

To improve Cleanifier further, future work could address new sampling schemes, such as more advanced skipping heuristics, minimizers [30] or syncmers [31]. Furthermore, a new human index may consider masking genomic regions that are known to have a high similarity with microbial sequences to enhance microbiome retention.

In addition, Cleanifier supports to build a custom database for other host organisms, e.g., to remove mouse reads from mouse gut microbiome samples. Further work could also investigate the effect of building a custom patient database if both whole genome sequencing and microbiome analyses are available from the same patient.

Code Availability

The code and Snakemake [32] workflows to reproduce all results are available via GitLab (`https://gitlab.com/rahmannlab/cleanifier`). In addition, Cleanifier is available at PyPi (`https://pypi.org/project/cleanifier/`) and Bioconda (`https://anaconda.org/bioconda/cleanifier`).

Data Availability

All data used in the evaluation is available online [33, 25]. Detailed accession numbers are provided in the Supplementary Material. The pre-built Cleanifier index is available at Zenodo `https://doi.org/10.5281/zenodo.15639519` or can be downloaded using the `cleanifier download` command.

References

1. Joana Pereira-Marques, Anne Hout, Rui M. Ferreira, Michiel Weber, Ines Pinto-Ribeiro, Leen-Jan van Doorn, Cornelis Willem Knetsch, and Ceu Figueiredo. Impact of Host DNA and Sequencing Depth on the Taxonomic Resolution of Whole Metagenome Sequencing for Microbiome Analysis. *Frontiers in Microbiology*, 10, 2019. Publisher: Frontiers.
2. Yuichi Kodama, Martin Shumway, Rasko Leinonen, and on behalf of the International Nucleotide Sequence Database

Collaboration. The sequence read archive: explosive growth of sequencing data. *Nucleic Acids Research*, 40:D54–D56, 2012.

3. Lita M. Proctor, Heather H. Creasy, Jennifer M. Fettweis, Jason Lloyd-Price, Anup Mahurkar, Wenyu Zhou, Gregory A. Buck, Michael P. Snyder, Jerome F. Strauss, George M. Weinstock, Owen White, Curtis Huttenhower, and The Integrative HMP (iHMP) Research Network Consortium. The Integrative Human Microbiome Project. *Nature*, 569(7758):641–648, 2019. Publisher: Nature Publishing Group.
4. Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, 2012. Publisher: Nature Publishing Group.
5. Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.
6. Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.
7. The Huttenhower Lab. KneadData – The Huttenhower Lab.
8. Bede Constantinides, Martin Hunt, and Derrick W Crook. Hostile: accurate decontamination of microbial host sequences. *Bioinformatics*, 39(12):btad728, 2023.
9. Derrick E. Wood, Jennifer Lu, and Ben Langmead. Improved metagenomic analysis with Kraken 2. *Genome Biology*, 20(1):257, 2019.
10. F. P. Breitwieser, D. N. Baker, and S. L. Salzberg. KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome Biology*, 19(1):198, 2018.
11. Kenneth S. Katz, Oleg Shutov, Richard Lapoint, Michael Kimelman, J. Rodney Brister, and Christopher O'Sullivan. STAT: a fast, scalable, MinHash-based k-mer tool to assess Sequence Read Archive next-generation sequence submissions. *Genome Biology*, 22(1):270, 2021.
12. Sergey Nurk, Sergey Koren, Arang Rhie, et al. The complete sequence of a human genome. *Science*, 376(6588):44–53, 2022. Publisher: American Association for the Advancement of Science.
13. Wen-Wei Liao, Mobin Asri, Jana Ebler, et al. A draft human pangenome reference. *Nature*, 617(7960):312–324, 2023. Publisher: Nature Publishing Group.
14. Adam Auton, Gonçalo R. Abecasis, David M. Altshuler, et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015. Publisher: Nature Publishing Group.
15. Dominic J Barker, Giuseppe Maccari, Xenia Georgiou, Michael A Cooper, Paul Flicek, James Robinson, and Steven G E Marsh. The IPD-IMGT/HLA database. *Nucleic Acids Research*, 51:D1053–D1060, 2023.
16. Michael B Hall and Lachlan J M Coin. Pangenome databases improve host removal and mycobacteria classification from clinical metagenomic data. *GigaScience*, 13:giae010, 2024.
17. Jennifer Lu, Natalia Rincon, Derrick E. Wood, Florian P. Breitwieser, Christopher Pockrandt, Ben Langmead, Steven L. Salzberg, and Martin Steinegger. Metagenome analysis using the Kraken software suite. *Nature Protocols*, 17(12):2815–2839, 2022. Publisher: Nature Publishing Group.
18. Karel Břinda, Maciej Sykulski, and Gregory Kucherov. Spaced seeds improve k-mer-based metagenomic classification. *Bioinformatics*, 31(22):3584–3592, 2015.
19. Jens Zentgraf and Sven Rahmann. Fast gapped k-mer counting with subdivided multi-way bucketed cuckoo hash tables. In Christina Boucher and Sven Rahmann, editors, *22nd International Workshop on Algorithms in Bioinformatics (WABI 2022)*, volume 242 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. ISSN: 1868-8969.
20. Sven Rahmann and Jens Zentgraf. Design of Worst-Case-Optimal Spaced Seeds, 2023. Pages: 2023.11.20.567826 Section: New Results.
21. Jens Zentgraf and Sven Rahmann. Fast lightweight accurate xenograft sorting. *Algorithms for Molecular Biology*, 16(1):2, 2021.
22. Stefan Walzer. Load Thresholds for Cuckoo Hashing with Overlapping Blocks. *ACM Trans. Algorithms*, 19(3):24:1–24:22, 2023.
23. Johanna Elena Schmitz, Jens Zentgraf, and Sven Rahmann. Smaller and More Flexible Cuckoo Filters, 2025.
24. Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: a LLVM-based python JIT compiler. In Hal Finkel, editor, *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM 2015, Austin, TX*, pages 7:1–7:6. ACM, 2015.
25. Justin M. Zook, David Catoe, Jennifer McDaniel, et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific Data*, 3(1):160025, 2016. Publisher: Nature Publishing Group.
26. Fernando Meyer, Adrian Fritz, Zhi-Luo Deng, et al. Critical Assessment of Metagenome Interpretation: the second round of challenges. *Nature Methods*, 19(4):429–440, 2022. Publisher: Nature Publishing Group.
27. Adrian Fritz, Alice McHardy, Till Lesker, and Andreas Bremges. CAMI 2 - Multisample Benchmark Dataset of Human Microbiome Project. 2021. Publisher: Helmholtz-Zentrum für Infektionsforschung.
28. Curtis Huttenhower, Dirk Gevers, Rob Knight, et al. Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402):207–214, 2012. Publisher: Nature Publishing Group.
29. Yunyun Gao, Hao Luo, Hujie Lyu, Haifei Yang, Salsabeel Yousuf, Shi Huang, and Yong-Xin Liu. Benchmarking short-read metagenomics tools for removing host contamination. *GigaScience*, 14:giaf004, 2025.
30. Michael Roberts, Wayne Hayes, Brian R. Hunt, Stephen M. Mount, and James A. Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.
31. Robert Edgar. Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ*, 9:e10805, 2021. Publisher: PeerJ Inc.
32. F Mölder, KP Jablonski, B Letcher, MB Hall, CH Tomkins-Tinch, V Sochat, J Forster, S Lee, SO Twardziok, A Kanitz, A Wilm, M Holtgrewe, S Rahmann, S Nahnsen, and J Köster. Sustainable data analysis with Snakemake [version 2; peer review: 2 approved]. *F1000Research*, 10(33), 2021.
33. Fernando Meyer, Gary Robertson, Zhi-Luo Deng, David Koslicki, Alexey Gurevich, and Alice C McHardy. CAMI benchmarking portal: online evaluation and ranking of metagenomic software. *Nucleic Acids Research*, page gkaf369, 2025.

Supplementary Material

Data accession

	Short reads	Long reads
HG002	ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG002_NA24385_son/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG002-23100077/MPHG002_S1_L001_R1_001.fastq.gz, ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG002_NA24385_son/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG002-23100077/MPHG002_S1_L001_R2_001.fastq.gz	https://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data_indexes/AshkenazimTrio/sequence.index.AJtrio_PacBio_CCS_10kb_09032018.HG002 samples IDs: 54238_180628_014238, 54238_180629_191119, 54238_180630_152934, 54238_180701_114913, 54315_180629_192151
HG003	ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG003_NA24149_father/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG003-23100078/MPHG003_S2_L002_R1_001.fastq.gz, ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG003_NA24149_father/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG003-23100078/MPHG003_S2_L002_R2_001.fastq.gz	
HG004	ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG004_NA24143_mother/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG004-23100079/MPHG004_S3_L003_R1_001.fastq.gz, ftp://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG004_NA24143_mother/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG004-23100079/MPHG004_S3_L003_R2_001.fastq.gz	
HG005	ftp://ftp-trace.ncbi.nih.gov/ReferenceSamples/giab/data/ChineseTrio/HG005_NA24631_son/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG005-23100080/MPHG005_S4_L004_R1_001.fastq.gz, ftp://ftp-trace.ncbi.nih.gov/ReferenceSamples/giab/data/ChineseTrio/HG005_NA24631_son/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG005-23100080/MPHG005_S4_L004_R2_001.fastq.gz	
HG006	ftp://ftp-trace.ncbi.nih.gov/ReferenceSamples/giab/data/ChineseTrio/HG006_NA24694-huCA017E_father/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG006-23100081/MPHG006_S5_L005_R1_001.fastq.gz, ftp://ftp-trace.ncbi.nih.gov/ReferenceSamples/giab/data/ChineseTrio/HG006_NA24694-huCA017E_father/NIST_Stanford_Illumina_6kb_matepair/fastqs/MPHG006-23100081/MPHG006_S5_L005_R2_001.fastq.gz	
Gastrointestinal	https://frl.publisso.de/data/frl:6425518/gastrooral/ sample IDs: 0, 1, 2, 3	https://frl.publisso.de/data/frl:6425518/gastrooral_pbsim/ sample IDs: 0, 1, 2, 3
Oral	https://frl.publisso.de/data/frl:6425518/gastrooral/ sample IDs: 6, 7, 8, 13	https://frl.publisso.de/data/frl:6425518/gastrooral_pbsim/ sample IDs: 6, 7, 8, 13
Airways	https://frl.publisso.de/data/frl:6425518/airskinurogenital/ sample IDs: 7, 8, 10, 11	https://frl.publisso.de/data/frl:6425518/airskinurogenital_pbsim/ sample IDs: 7, 8, 10, 11
Skin	https://frl.publisso.de/data/frl:6425518/airskinurogenital/ sample IDs: 1, 13, 14, 15	https://frl.publisso.de/data/frl:6425518/airskinurogenital_pbsim/ sample IDs: 1, 13, 14, 15
Urogenital	https://frl.publisso.de/data/frl:6425518/airskinurogenital/ sample IDs: 0, 2, 3, 6	https://frl.publisso.de/data/frl:6425518/airskinurogenital_pbsim/ sample IDs: 0, 2, 3, 6

Table 3. Data accesion for human (GIAB) and microbiome (CAMI 2 challenge) data. The data is subsampled (for human) or combined (for microbiome) such that each file contains 50 million short reads or 5 million long reads.

Cleanifier speedup

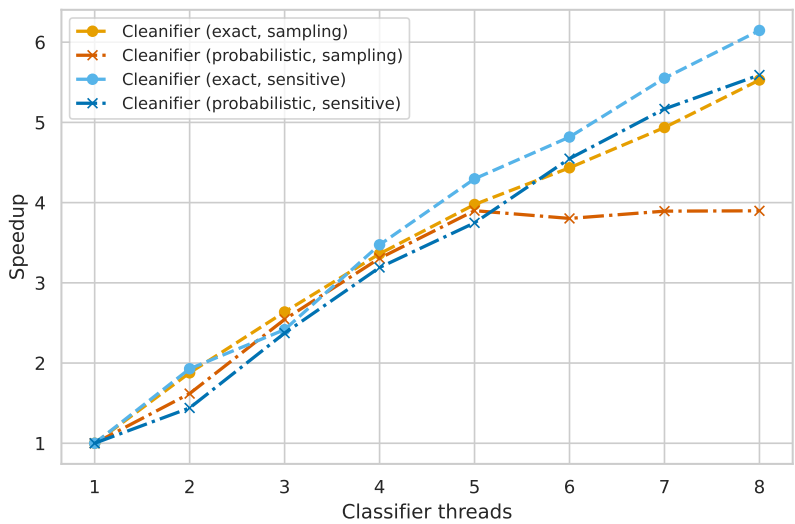

Fig. S.1: Speedup for increasing classifier threads. Total threads are the number of classifier threads plus one additional read and one additional write thread.

The speedup factor from parallelization is almost linear (Figure S.1) for up to 8 classifier threads except for Cleanifier (probabilistic, sampling). For Cleanifier (probabilistic, sampling) the read or write thread become the main bottleneck if more than 5 threads perform the classification.

Accuracy for different classification thresholds and k-mer masks

k	w	mask
23	33	`'####_##__#_####_#_####_#__##_####'`
27	31	`'######_#####_#####_#####_######'`
29	33	`'######_#######_###_#######_######'`
29	37	`'####_####_###_###_#_###_###_####_####'`
31	37	`'######_###_####_#####_####_###_######'`
31	43	`'####_#_##_###_#_###_###_###_#_###_##_#_####'`

Table 4. Evaluated gapped k-mer masks.

We built indexes for 6 different k-mer masks that were selected from the best performing k-mer masks evaluated in [20], see Table 4. Figure S.2 shows the accuracy for human and microbiome data for the different masks and different classification thresholds. Due to the good performance on both human and microbiome data, we selected the (29,33)-mask and set the default threshold for classification to 0.5.

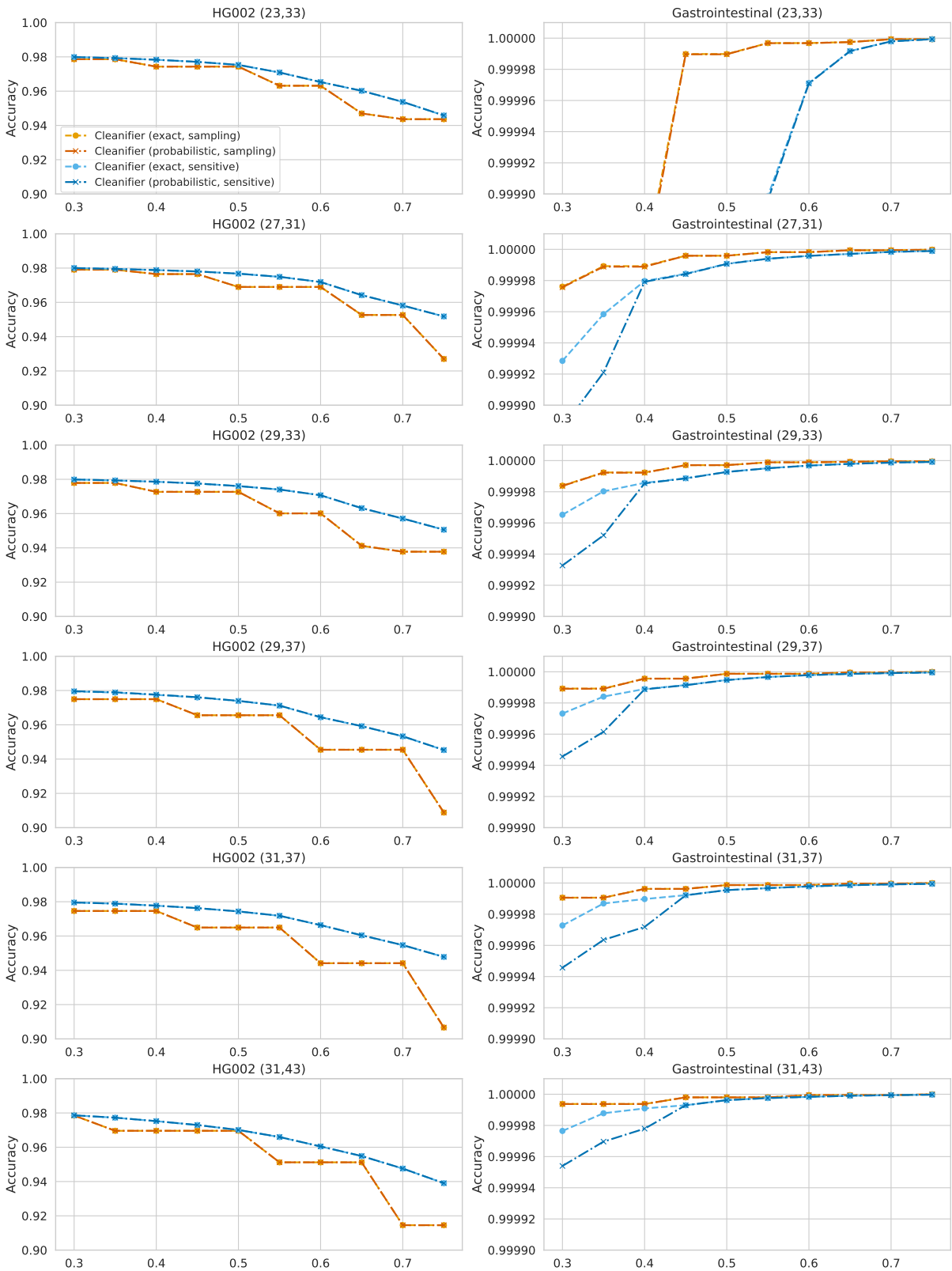

Fig. S.2: Accuracy for human and microbiome data for different indexes (built using different gapped k-mer masks) and different classification thresholds.

Comparison of T2T and the advanced human index

	Cleanifier exact				Cleanifier probabilistic			
	sampling		sensitive		sampling		sensitive	
	T2T	pangenome	T2T	pangenome	T2T	pangenome	T2T	pangenome
Human	0.969986	0.972677	0.975124	0.976031	0.969988	0.972680	0.975136	0.976045
Microbiome	0.999999	0.999997	0.999996	0.999993	0.999999	0.999997	0.999996	0.999993

Table 5. Accuracy for an index built for T2T reference and an index containing all gapped k-mers of the T2T reference, common variants, HLA variants and multiple pangenome assemblies. We measured the accuracy on the HG002 and gastrointestinal short read datasets.

Table 5 shows the accuracy for an index containing only the gapped k-mers from the T2T reference [12] and from our advanced human index that is built from the T2T reference, the common variants from the 1000 Genome Project [14], the HLA variants from the IPD-IMGT/HLA database [15] and the 47 assemblies from the Human Pangenome Consoritium [13]. Our advanced index leads to a higher accuracy on human data both for the exaxt and probabilistic version, with a higher increase in the accuracy for the sampling mode. The microbiome retention is only marginally effected by the advanced index.

Pairwise overlap between falsely classified human reads.

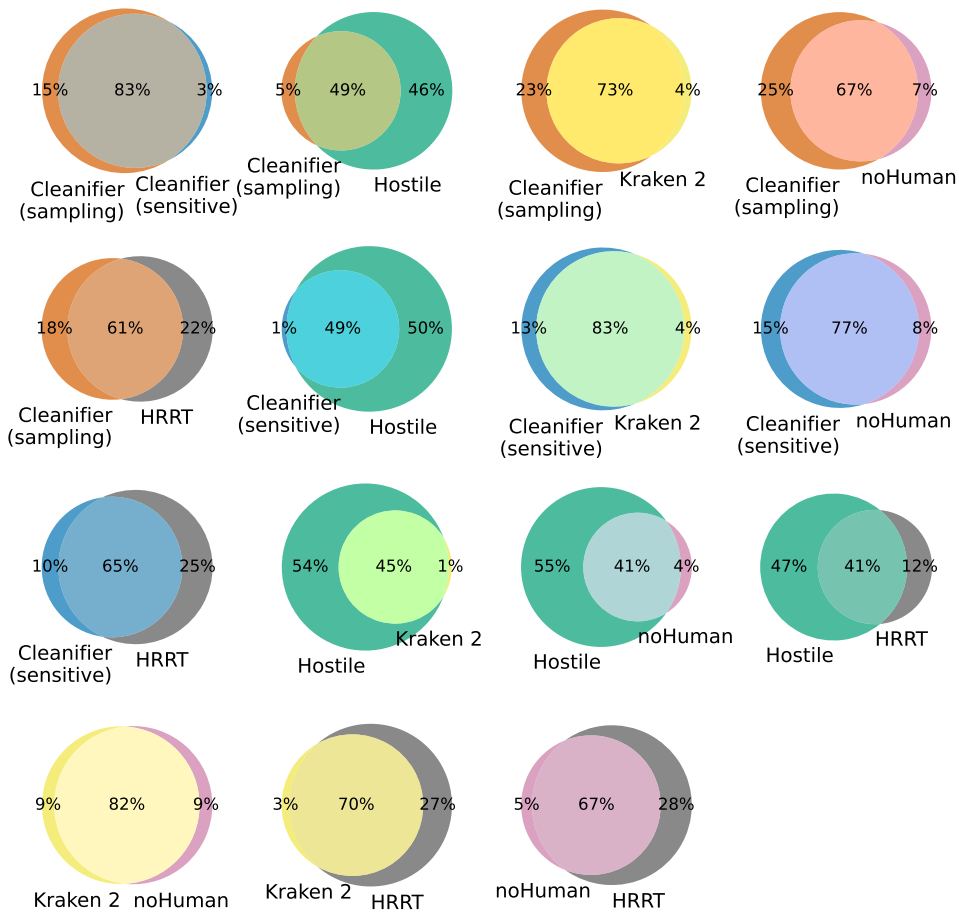

Fig. S.3: Overlap of falsely classified reads of the HG002 dataset between all tools. We only included Cleanifier (probabilisitc) in the comparison.

Figure S.3 shows the pairwise overlap of reads that are falsely classified in the HG002 dataset of all evaluated tools. Since all tools have a high overlap, it is possible that these reads are either contaminants, like PhiX or the Eppstein-Barr virus, or poor quality reads.