KrakenHLL: Confident and fast metagenomics classification using unique k-mer counts

Breitwieser FP[1] and Salzberg SL[1,2]

1 Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins School of Medicine, Baltimore, MD, United States

2 Departments of Biomedical Engineering, Computer Science and Biostatistics, Johns Hopkins University, Baltimore, MD, United States

Abstract

Motivation: False positive identifications are a significant problem in metagenomics. When matching sequencing reads to a database of genomes, low-complexity regions and contaminants in genome assemblies can attract many reads to the wrong genomes. Alignment information may be used to recognize that many reads are piling up in a small number of locations, but fast k-mer based metagenomic classifiers do not provide alignments, and re-alignment is expensive. We propose using k-mer coverage, which can be computed during classification, as a proxy for genome base coverage.

Results: We present KrakenHLL, a sequence classifier that records the number of unique k-mers observed in each taxon in a metagenomics data set. KrakenHLL is based on the ultra-fast classification engine Kraken, which it combines with a cardinality estimator known as HyperLogLog (HLL). We demonstrate that many false-positive identifications can be identified using unique k-mer counts, especially when looking at species present in low abundance. KrakenHLL's database includes over 100,000 additional viral strain sequences not used in the standard Kraken database, and it also has the ability to map against multiple databases and

identify plasmids and strains using an extended taxonomy. KrakenHLL's run time is similar to that of Kraken, and it requires only a very modest increase in memory.

Availability and Implementation: KrakenHLL is implemented in C++ and Perl, and available under the GPL v3 license at https://github.com/fbreitwieser/krakenhll.

Contact: florian.bw@gmail.com.

Introduction

Metagenomic classifiers attempt to assign a taxonomic identity to each read in a data set. Because metagenomics data often contain tens of millions of reads, classification is typically done using exact matching of short words of length *k* (*k*-mers) rather than alignment, which would be unacceptably slow. The results contain read classifications but not their aligned positions in the genomes (as reviewed by Breitwieser, et al., 2017). However, read counts can be deceiving. Sequence contamination of the samples–introduced from laboratory kits or the environment during sample extraction, handling or sequencing–can yield high numbers of spurious identifications (Salter, et al., 2014; Thoendel, et al., 2017). Having only small amounts of input material can further compound the problem of contamination. When using sequencing for clinical diagnosis of infectious diseases, for example, less than 0.1% of the DNA may derive from microbes of interest (Brown, et al., 2018; Salzberg, et al., 2016). Additional spurious matches can result from low-complexity regions of genomes and from contamination in the database genomes themselves (Mukherjee, et al., 2015).

Such false-positive reads typically match only small portions of a genome; e.g., if a species' genome contains a low-complexity region, and the only reads matching that species fall in this region, then the species was probably not present in the sample. Reads from microbes that are

47 truly present should distribute relatively uniformly across the genome rather than being
48 concentrated in one or a few locations. Genome alignment can reveal this information. However,
49 alignment is resource intensive, requiring the construction of indexes for every genome and a
50 relatively slow alignment step to compare all reads against those indexes. Some metagenomics
51 methods do use coverage information to improve mapping or quantification accuracy, but these
52 methods require results from much slower alignment methods as input (Dadi, et al., 2017).
53 Assembly-based methods also help to avoid false positives, but these are useful only for highly
54 abundant species (Quince, et al., 2017).
55
56 Here, we present KrakenHLL, a novel method that combines very fast k-mer based classification
57 with a fast k-mer cardinality estimation. KrakenHLL is based on the Kraken metagenomics
58 classifier (Wood and Salzberg, 2014), to which it adds a method for counting the number of
59 unique k-mers identified for each taxon using the efficient cardinality estimation algorithm
60 HyperLogLog (Ertl, 2017; Flajolet, et al., 2007; Heule, et al., 2013). By counting how many of
61 each genome's unique k-mers are covered by reads, KrakenHLL can often discern false positive
62 from true positive matches. Furthermore, KrakenHLL implements additional new features to
63 improve metagenomics classification: (a) searches can be done against multiple databases
64 hierarchically, (b) the taxonomy can be extended to include nodes for strains and plasmids, thus
65 enabling their detection, and (c) the database build script allows the addition of >100,000 viruses
66 from the NCBI Viral Genome Resource (Brister, et al., 2015). KrakenHLL provides a superset of
67 the information provided by Kraken while running equally fast or slightly faster, and while using
68 very little additional memory during classification.

Results

KrakenHLL was developed to provide efficient k-mer count information for all taxa identified in a metagenomics experiment. The main workflow is as follows: As reads are processed, each k-mer is assigned a taxon from the database (Figure 1A). KrakenHLL instantiates a HyperLogLog data sketch for each taxon, and adds the k-mers to it (Figure 1B and Supplementary Information). After classification of a read, KrakenHLL traverses up the taxonomic tree and merges the estimators of each taxon with its parent. In its classification report, KrakenHLL includes the number of unique k-mers and the depth of k-mer coverage for each taxon that it observed in the input data (Figure 1C).

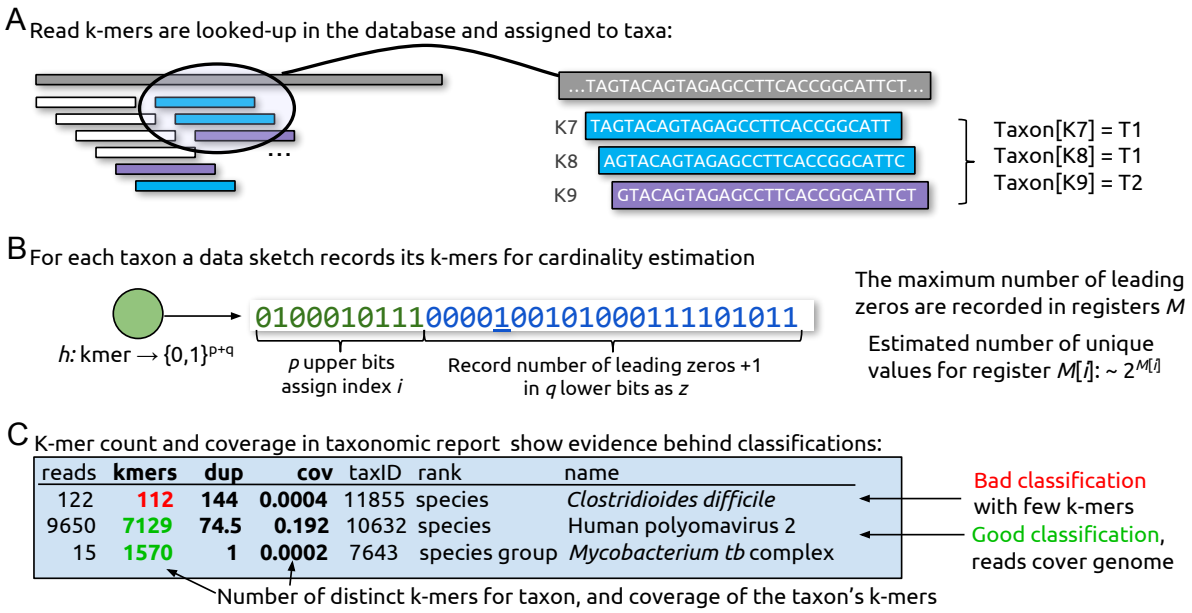


Figure 1. Overview of the KrakenHLL algorithm and output. (A) An input read is shown as a long gray rectangle, with k-mers shown as shorter rectangles below it. The taxon mappings for each k-mer are compared to the database, shown as larger rectangles on the right. For each taxon, a unique k-mer counter is instantiated, and the observed k-mers (K7, K8, and K9) are added to

the counters. (B) Unique k-mer counting is implemented with the probabilistic estimation method HyperLogLog (HLL) using 16KB of memory per counter, which limits the error in the cardinality estimate to 1% (see main text). (C) The output includes the number of reads, unique k-mers, duplicity (average time each k-mer has been seen) and coverage for each taxon observed in the input data.

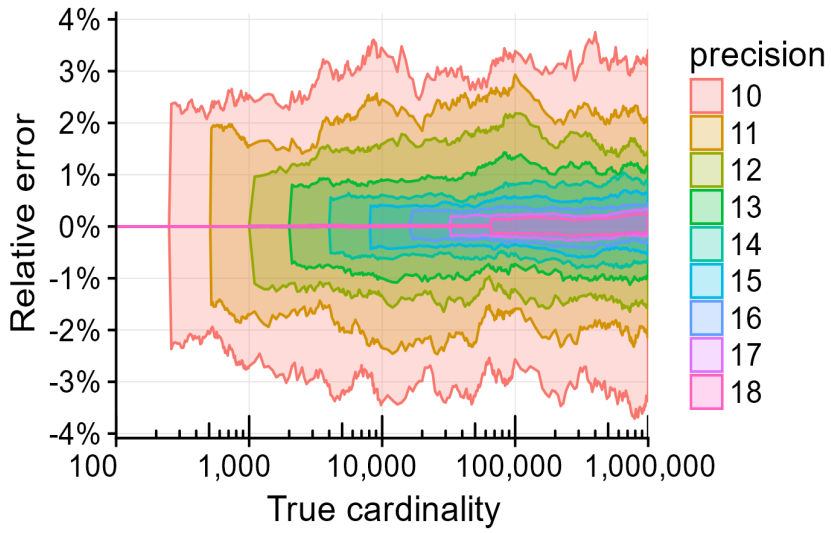
Efficient k-mer cardinality estimation using the HyperLogLog algorithm

Exact counting of the number of unique values (cardinality) of a string (for example, the number of instances of each word in a document) in the presence of duplicates requires memory proportional to the cardinality. Very accurate *estimation* of the cardinality, however, can be achieved using only a small, constant amount of space. The HyperLogLog algorithm (HLL), originally described by (Flajolet, et al., 2007), is currently one of the most efficient cardinality estimators and has recently been used to count k-mers in DNA sequences (Irber Junior and Brown, 2016). The main idea behind the method is that long runs of leading zeros are unlikely in random bitstrings. E.g., in a series of random bitstrings (or hashes) we expect to see every fourth bitstring start with exactly one 0-bit before the first 1-bit (01_2), and every 32nd hash starts with 00001_2. The HLL algorithm saves a "sketch" of observed data based on hashes of the k-mers in 2^p one-byte registers (in our implementation), where p is the precision parameter. The relative error of the estimate is 1/sqrt(2^p). With p=14, the sketch uses 2^{14} one-byte registers, i.e. 16 KB of space, and has a relative error less than 1% (Figure 2).

Generating the sketch. Each k-mer is first hashed into a 64-bit string H. The sketch starts out in a sparse representation which has an effective p of 25, using 4 bytes per element. See (Heule, et

al., 2013) for more details on the encoding. Once $m/4$ distinct elements have been observed, we switch to the standard representation of (Flajolet, et al., 2007): the first p bits of H are used as index i into the registers M. The remaining $64-p=q$ bits are used to define the rank based on the position of the first 1-bit (or, equivalently, the count of leading zeros plus one). If all q bits are zero, the rank is $q+1$. The register $M[i]$ is updated if the rank is higher than the current value of $M[i]$.

When the read classification is finished, KrakenHLL aggregates the taxon sketches up the taxonomy tree. Each taxon's sketch is merged with its children's sketches. The cardinality estimate is computed using a recently described method (Ertl, 2017) that does not require empirically determined thresholds to account for biases and switching between linear counting and the HLL estimator (Suppl. Figures 1 and 2). Figure 2 shows the performance and memory usage of KrakenHLL's cardinality estimator for up to one million k-mers. A detailed description of the algorithm and implementation is in the Supplementary Materials.



p	m	Space (kB)	Rel. Error
10	1024	1	3.25%
11	2048	2	2.23%
12	4096	4	1.63%
13	8192	8	1.15%
14	16384	16	0.81%
15	32768	32	0.57%
16	65536	64	0.41%
17	131072	128	0.29%
18	262144	256	0.20%
25			0.02%

123 Figure 2: Cardinality estimation using HyperLogLog for randomly sampled k-mers from
124 microbial genomes. Left: standard deviations of the relative errors of the estimate with precision
125 p ranging from 10 to 18. No systematic biases are apparent, and, as expected, the errors decrease
126 with higher values of p. Up to cardinalities of about $2^p/4$, the relative error is near zero. At higher
127 cardinalities, the error boundaries stay near constant. Right: the size of the registers, space
128 requirement, and expected relative error for HyperLogLog cardinality estimates with different
129 values of p. For example, with a precision p=14, the expected relative error is 0.81% and the
130 counter only requires 16 KB of space, which is three orders of magnitude less than that of an
131 exact counter (at a cardinality of one million). Up to cardinalities of $2^p/4$, KrakenHLL uses a
132 sparse representation of the counter with a higher precision of 25 and an effective relative error
133 rate of about 0.02%.
134

135 **Results on twenty-one simulated and ten biological test datasets**

136 We assessed KrakenHLL's performance on the 34 datasets compiled by (McIntyre, et al., 2017)
137 (see Suppl. Table 3). We place greater emphasis on the eleven biological datasets, which contain
138 more realistic laboratory and environmental contamination. In the first part of this section, we
139 show that unique k-mer counts provide classification accuracy than read counts, and in the
140 second part we compare KrakenHLL with the results of eleven metagenomics classifiers
141 described in (McIntyre, et al., 2017). We ran KrakenHLL on two databases: 'std', which includes
142 the human genome plus the complete bacterial, archaeal and viral genomes from RefSeq, and
143 'nt', which contains the microbial sequences in the BLAST nt sequence set (see Suppl. Methods
144 Section 2). We use two metrics to compare how well methods can separate true positives and
145 false positives: the maximum F1 score and maximum recall. Higher values of F1 (the harmonic

146 mean of precision and recall) indicate a better separation between true positives and false
147 positives. Higher recall (at the fixed false discovery rate of 0.05) means that more true species
148 can be recovered while controlling the false positives4.
149
150 Because the NCBI taxonomy has been updated since the datasets were published, we manually
151 updated the "truth" sets in several datasets (see Suppl. Methods Section 2.3 for all changes). Any
152 cases that might have been missed would result in a lower apparent performance of KrakenHLL.
153 Note that we exclude the over ten-year-old simulated datasets simHC, simMC and simLC from
154 Mavromatis et al. (2007), as well as the biological dataset JGI SRR033547 which has only 100
155 reads.
156

			orig			std			nt		
Data Type	Rank	Statistic	Reads	Kmers	Diff	Reads	Kmers	Diff	Reads	Kmers	Diff
Biological	Genus	Recall	0.90	0.93	**+4.0%**	0.89	0.94	**+6.2%**	0.91	0.99	**+8.9%**
Biological	Genus	F1	0.95	0.96	**+1.8%**	0.95	0.97	**+2.6%**	0.96	0.99	**+3.4%**
Biological	Species	Recall	0.85	0.87	**+2.6%**	0.70	0.78	**+11.8%**	0.95	0.98	**+3.1%**
Biological	Species	F1	0.94	0.94	**+0.7%**	0.90	0.92	**+2.5%**	0.97	0.99	**+1.6%**
Simulated	Genus	Recall	0.96	0.94	-2.1%	0.95	0.97	**+2.5%**	0.98	0.99	**+0.8%**
Simulated	Genus	F1	0.98	0.98	-0.0%	0.98	0.98	**+0.3%**	0.99	0.99	**+0.3%**
Simulated	Species	Recall	0.92	0.93	**+0.6%**	0.88	0.88	**+0.3%**	0.90	0.90	-0.1%
Simulated	Species	F1	0.97	0.97	**+0.3%**	0.94	0.94	**+0.5%**	0.96	0.96	-0.1%

157
158 Table 1: Performance of read count and unique k-mer thresholds on 10 biological and 21
159 simulated datasets against three databases (‘orig’, ‘std’, ‘nt). Unique k-mer count thresholds give
160 up to 10% better recall and F1 scores on biological datasets.
161
162 ***Unique k-mer versus read count thresholds***

163 We first looked at the performance of the unique k-mer count thresholds versus read count
164 thresholds (as would be used with Kraken). The k-mer count thresholds worked very well,
165 particularly for the biological datasets (Table 1 and Suppl. Table 3). On the genus level, the
166 average recall in the biological datasets increases by 4-9%, and the average F1 score increases 2-
167 3%. On the species level, the average increase in recall in the biological sets is between 3 and
168 12%, and the F1 score increases by 1-2%.
169
170 On the simulated datasets, the differences are less pronounced and vary between databases, even
171 though on average the unique k-mer count is again better. However, only in two cases (genus
172 recall on databases 'orig' and 'std') the difference is higher than 1% in any direction. We find
173 that simulated datasets often lack false positives with a decent number of reads but a lower
174 number of unique k-mer counts, which we see in real data. Instead, in most simulated datasets
175 the number of unique k-mers is linearly increasing with the number of unique reads in both true
176 and false positives (Suppl. Figure 4). In biological datasets, sequence contamination and lower
177 read counts for the true positives make the task of separating true and false positives harder.
178
179 ***Comparison of KrakenHLL with eleven other methods***
180 Next, we compared KrakenHLL's unique k-mer counts with the results of eleven metagenomics
181 classifiers from (McIntyre, et al., 2017). KrakenHLL with database 'nt' has the highest average
182 recall and F1 score across the biological datasets, as shown in Table 2. As seen before, using
183 unique k-mer instead of read counts as thresholds increases the scores. While the database
184 selection proves to be very important (KrakenHLL with database 'std' is performing 10% worse
185 than KrakenHLL with database 'nt'), only Blast has higher average scores than KrakenHLL with

186 k-mer count thresholds on the original database. On the simulated datasets, KrakenHLL with the
187 'nt' database still ranks at the top, though, as seen previously there is more variation (Suppl.
188 Table 4). Notably CLARK is as good as KrakenHLL, but Blast has much worse scores on the
189 simulated datasets.
190

	Genus		Species		
	F1	Recall	F1	Recall	Avg
KrakenHLL nt kmers	**0.99**	**0.99**	**0.99**	**0.98**	**0.99**
KrakenHLL nt reads	0.96	0.91	0.97	0.95	0.95
BlastMeganFilteredLiberal	0.97	0.94	0.97	0.89	0.94
BlastMeganFiltered	0.97	0.93	0.96	0.87	0.93
KrakenHLL orig kmers	0.96	0.93	0.94	0.87	0.93
ClarkM4Spaced	0.95	0.90	0.94	0.88	0.92
KrakenHLL orig reads	0.95	0.90	0.94	0.85	0.91
Kraken	0.95	0.90	0.94	0.84	0.91
KrakenHLL std kmers	0.97	0.94	0.92	0.78	0.90
DiamondMegan_sensitive	0.98	0.93	0.92	0.74	0.89
KrakenFiltered	0.95	0.91	0.90	0.75	0.88
ClarkM1Default	0.94	0.85	0.91	0.77	0.87
KrakenHLL std reads	0.95	0.89	0.90	0.70	0.86
LMAT	0.97	0.93	0.91	0.60	0.85
DiamondMegan	0.94	0.87	0.91	0.66	0.85
Gottcha	0.91	0.84	0.87	0.67	0.82
NBC	0.87	0.76	0.85	0.73	0.80
Metaphlan	0.94	0.89	0.83	0.55	0.80
MetaFlow	0.66	0.53	0.65	0.51	0.59
PhyloSift	0.68	0.29	0.78	0.54	0.57
PhyloSift90pct	0.68	0.30	0.77	0.52	0.57

191

192 Table 2: Performance of KrakenHLL (with unique k-mer count thresholds) compared to
193 metagenomic classifiers used in (McIntyre, et al., 2017) on the biological datasets (n=10). F1 and
194 Recall show the average values over the datasets. Note that 'KrakenHLL reads' would be
195 equivalent to standard Kraken.

196

197

198 **Generating a better test dataset, and selecting an appropriate k-mer threshold**

199 We demonstrated that, given good thresholds, KrakenHLL performs better than other classifiers
200 on the test datasets. The test datasets, however, are not ideal for determining guidelines for
201 thresholds on real data. To establish thresholds, we'd like to use biological data, but the lab-
202 created microbial communities which are the basis of the datasets are usually of very low
203 complexity. For example, only two of the biological test datasets of the previous sections had
204 more than ten species. While some of the simulated test datasets have a much higher complexity,
205 they lack features of biological dataset.

206

207 We propose a third type of test dataset generated by mixing reads from isolate sequencing
208 experiments (of which there are tens of thousands). By subsampling the isolate experiments, we
209 can create an arbitrarily complex test dataset for which we know the ground truth. All reads are
210 from actual experiments, and we may contain contaminants and sequencing errors. We selected
211 280 SRA datasets from 280 different bacterial species that are linked to complete RefSeq
212 genomes (see Suppl. Methods Section 2.4). We randomly sampled between one hundred and one
213 million reads (logarithmically distributed) from each species, which gave 34 million read pairs in
214 total. To see the effect of sequencing depth, we created five sub-sampled read sets with one to
215 twenty million reads, too. All read sests were classified with KrakenHLL with the 'std' database.

216

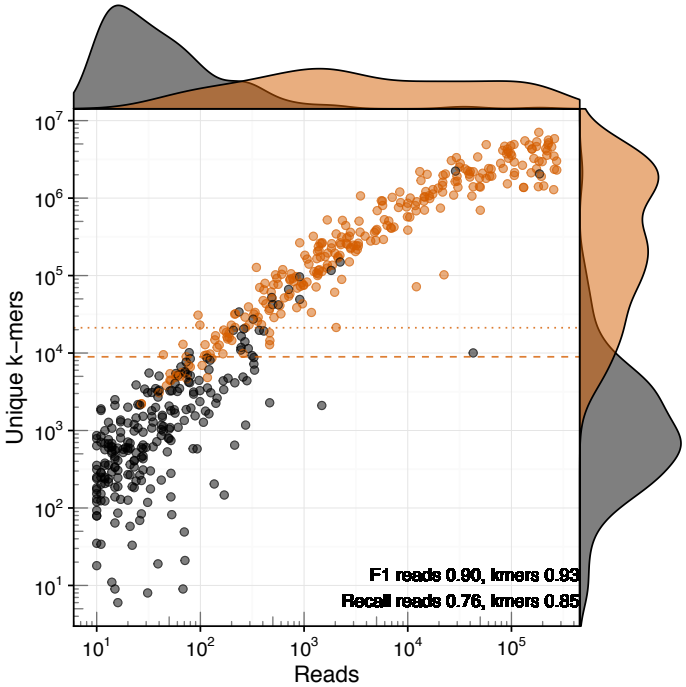
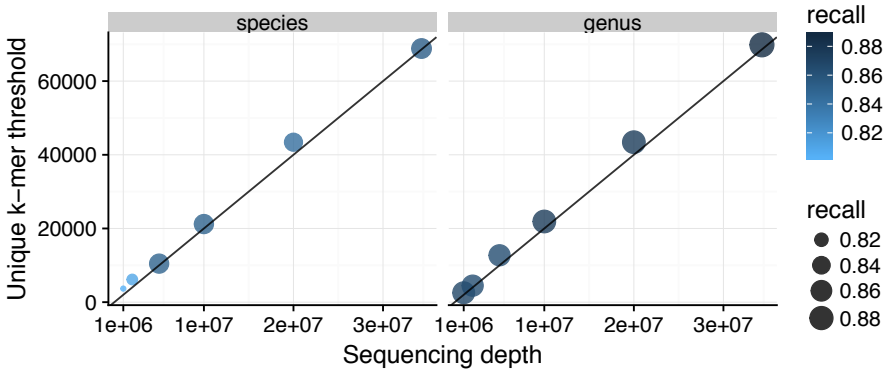


Figure 3: Unique k-mer count separates true and false positives better than read counts in a complex dataset with ten million reads sampled from SRA experiments. Each dot represents a species, with true species in orange and false species in black. The dashed line shows the k-mer threshold giving the best F1 score, and the dotted line shows the threshold for the best recall at a maximum of 5% FDR. In this dataset, a unique k-mer count in the range 10000–20000 would give the best threshold for selecting true species.

Consistent with the results of the results of the previous section, we found that unique k-mer counts provide better thresholds than read counts both in terms of F1 score and recall in all test datasets (e.g. Figure 3 on ten million reads – species recall using k-mers is 0.85, recall using reads 0.76). With higher sequencing depth, the recall increased slightly - from 0.80 to 0.85 on the species level, and from 0.87 to 0.89 on the genus level. The ideal values of the unique k-mer count thresholds, however, vary widely with different sequencing depths. We found that the ideal

231 thresholds increase by about 2000 unique k-mers per one million reads (see Figure 4). (McIntyre,
232 et al., 2017) found that k-mer based methods show a positive relationship between sequencing
233 depths and misclassified reads. Our analysis suggest that the sequencing depth definitely should
234 be taken into into account. Note that Bracken may help reduce false positive identification of
235 similar species in the same genus, because it can redistribute reads from species at low counts to
236 species at higher abundance (Lu, et al., 2017).
237



238

No. of reads	Fraction	Genus Threshold	Genus Recall	Species Threshold	Species Recall
1 million	0.03	2555	0.87	3682	0.80
2 million	0.06	4483	0.86	6152	0.81
5 million	0.15	12723	0.87	10459	0.85
10 million	0.3	21896	0.88	21201	0.85
20 million	0.6	43417	0.88	43417	0.84
34.3 million	1	69847	0.89	688842	0.85

239

240 Figure 4: Optimal thresholds vary with sequencing depth. A minimum threshold of about 2000
241 unique k-mer per a million reads gives the best results in this dataset (solid line in plot).
242

243 In general, we find that for correctly identified species, we obtain up to approximately *L-k*
244 unique k-mers per each read, where *L* is the read length, because each read samples a different
245 location in the genome. (Note that once the genome is completely covered, no more unique k-

246 mers can be detected.) Thus the k-mer threshold should always be several times higher than the
247 read count threshold. For the discovery of pathogens in human patients, discussed in the next
248 section, a read count threshold of 10 and unique k-mer count threshold of 1000 eliminated many
249 background identifications while preserving all true positives, which were discovered from as
250 few as 15 reads.
251

252 **Results on biological samples for infectious disease diagnosis**

253 Metagenomics is increasingly used to find species of low abundance. A special case is the
254 emerging use of metagenomics for the diagnosis of infectious diseases (Simner, et al., 2017;
255 Zhang, et al., 2015). In this application, infected human tissues are sequenced directly to find the
256 likely disease organism. Usually, the vast majority of the reads match (typically 95-99%) the
257 host, and sometimes fewer than 100 reads out of many millions of reads are matched to the target
258 species. Common skin bacteria from the patient or lab personnel and other contamination from
259 sample collection or preparation can easily generate a similar number of reads, and thus mask the
260 signal from the pathogen.
261

262 To assess if the unique k-mer count metric in KrakenHLL could be used to rank and identify
263 pathogen from human samples, we reanalyzed ten patient samples from a previously described
264 series of neurological infections (Salzberg, et al., 2016). That study sequenced spinal cord mass
265 and brain biopsies from ten hospitalized patients for whom routine tests for pathogens were
266 inconclusive. In four of the ten cases, a likely diagnosis could be made with the help of
267 metagenomics. To confirm the metagenomics classifications, the authors in the original study re-
268 aligned all pathogen reads to individual genomes.

269

270 Table 3 shows the results of our reanalysis of the confirmed pathogens in the four patients,
271 including the number of reads and unique k-mers from the pathogen, as well as the number of
272 bases covered by re-alignment to the genomes. Even though the read numbers are very low in
273 two cases, the number of unique k-mers suggests that each read matches a different location in
274 the genome. For example, in PT8, 15 reads contain 1570 unique k-mers, and re-alignment shows
275 2201 covered base pairs. In contrast, Table 4 shows examples of identifications from the same
276 datasets that are not well-supported by k-mer counts.

277

278 Table 3: Pathogen identifications in patients with neurological infections. The pathogens were
279 identified with as few as 15 reads, but those mapped to a high number of unique k-mers,
280 indicating distinct locations of the reads along their genomes. "Bases" refers to the number of
281 bases covered by the reads after re-alignment to a selected genome. Interestingly, the k-mer
282 count in PT5 reveals that there seems to be more than one viral strain present, as the k-mers
283 cover more than one genome. Read lengths were 150-250 bp.

Sample	Name	Reads	K-mers	Bases
PT5	Human polyomavirus 2	9650	7129*	5130
PT7	*Elizabethkingia genomosp. 3*	403	20724	52921
PT8	*Mycobacterium tuberculosis*	15	1570	2201
PT10	Human gammaherpesvirus 4	20	2084	2780

284

285 Table 4: False positive identifications have few unique k-mers per read. Note that the viral
286 identifications in PT4 and PT10 come from the NCBI Viral Genome Resource. Since the
287 KrakenHLL reports sequence-level matches, the source genomes are easy to find.

Sample	Name	Reads	K-mers
PT3	*Clostridioides difficile*	122	126
PT4	Hepatitis C virus JF343788.1 Recombinant Hepatitis C virus	101	3
PT5	*Akkermansia muciniphila*	936	136
PT10	Human betaherpesvirus 5 JN379815.1 Human herpesvirus 5 strain U04, partial genome	63	5

288

289

290 **Further extensions in KrakenHLL**

291 KrakenHLL adds three further notable features to the classification engine.

292 1. Enabling strain identification by extending the taxonomy: The finest level of granularity
293 for Kraken classifications are nodes in the NCBI taxonomy. This means that many strains
294 cannot be resolved, because up to hundreds of strains share the same taxonomy ID.
295 KrakenHLL allows extending the taxonomy with virtual nodes for genomes,
296 chromosomes and plasmids, and thus enabling identifications at the most specific levels
297 (see Suppl. Methods Section 3)
298 2. Integrating 100,000 viral strain sequences: RefSeq includes only one reference genome
299 for most viral species, which means that a lot of the variation of viral strain is not covered
300 in a standard RefSeq database. KrakenHLL sources viral strain sequences from the NCBI

301 Viral Genome Resource that are validated as ‘neighbors’ of RefSeq viruses, which leads
302 to up to 20% more read classifications (see Suppl. Methods Section 4).
303 3. Hierarchical classification with multiple databases. Often, we would like to include
304 additional sequence sets in some searches - for example draft genomes that may contain
305 contamination. KrakenHLL allows to chain databases in classification and matches each
306 k-mer hierarchically, stopping after the first match. For example, to mitigate the problem
307 of host contamination in draft genomes, a search may use the host genome as first
308 database, then the complete microbial genomes, then draft microbial genomes. More
309 details are available in Suppl. Method Section 5.
310

311 **Timing and memory requirements**

312 The additional features of KrakenHLL come without a runtime penalty and very limited
313 additional memory requirements. In fact, due to code improvements, KrakenHLL often runs
314 faster than Kraken, particularly when most of the reads come from one species. On the test
315 dataset, the mean classification speed in million base-pairs per minute increased slightly from
316 410 to 421 Mbp/m (see Suppl. Table 3). When factoring in the time needed to summarize
317 classification results by kraken-report, which is required for Kraken but part of the classification
318 binary of KrakenHLL, KrakenHLL is on average 50% faster. The memory requirements increase
319 on average by 0.5 GB from 39.5 GB to 40 GB.
320

321 On the pathogen Id patient data, where in most cases over 99% of the reads were either assigned
322 to human or synthetic reads, KrakenHLL was significantly faster than Kraken (Suppl. Table 5).
323 The classification speed increased from 467 to 733 Mbp/m. The average wall time was about

44% lower, and the average additional memory requirements were less than 1GB, going from 118.0 to 118.4 GB. All timing comparisons were made after preloading the database and running with 10 parallel threads.

Conclusions

KrakenHLL is a novel method that combines fast k-mer based classification with an efficient algorithm for counting the number of unique k-mers found in each species in a metagenomics dataset. When the reads from a species yield many unique k-mers, one can be more confident that the taxon is truly present, while a low number of unique k-mers suggests a possible false positive identification. We demonstrated that using unique k-mer counts provides improved accuracy for species identification, and that k-mer counts can help greatly in identifying false positives. In our comparisons with multiple other metagenomics classifiers on multiple metagenomics datasets, we found that KrakenHLL consistently ranked at the top. The strategy of counting unique k-mer matches allows KrakenHLL to detect that reads are spread across a genome, without the need to align the reads. By using a probabilistic counting algorithm, KrakenHLL is able to match the exceptionally fast classification time of the original Kraken program with only a very small increase in memory. The result is that KrakenHLL gains many of the advantages of alignment at a far lower computational cost.

Acknowledgements

Thanks to Jen Lu, Ales Varabyou, David Karig, Susan Bewick, Peter Thielen, and Thomas Mehoke for valuable discussions. Thanks to Daniel Baker for discussions and suggestions on improvements regarding the HyperLogLog algorithm, and to Jessica Atwell for proofreading the

manuscript. We further would like to thank Alexa McIntyre and Rachid Ounit for providing the scripts for their benchmarking paper, very quick answers to all my questions, and providing the Kraken database used in the analysis.

Funding

This work was supported in part by grants R01-GM083873 and R01-HG006677 from the National Institutes of Health, and by grant number W911NF-14-1-0490 from the U. S. Army Research Office.

References

Breitwieser, F.P., Lu, J. and Salzberg, S.L. A review of methods and databases for metagenomic classification and assembly. *Brief Bioinform* 2017.

Brister, J.R.*, et al.* NCBI Viral Genomes Resource. *Nucleic Acids Research* 2015;43(D1):D571-D577.

Brown, J.R., Bharucha, T. and Breuer, J. Encephalitis diagnosis using metagenomics: application of next generation sequencing for undiagnosed cases. *Journal of Infection* 2018.

Dadi, T.H.*, et al.* SLIMM: species level identification of microorganisms from metagenomes. *PeerJ* 2017;5:e3138.

Ertl, O. New Cardinality Estimation Methods for HyperLogLog Sketches. *arXiv:1706.07290* 2017.

Flajolet, P.*, et al.* HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In, *AofA: Analysis of Algorithms*. Juan les Pins, France: Discrete Mathematics and Theoretical Computer Science; 2007. p. 137-156.

Heule, S., Nunkesser, M. and Hall, A. HyperLogLog in practice. 2013:683.

Irber Junior, L.C. and Brown, C.T. Efficient cardinality estimation for k-mers in large DNA sequencing data sets. *bioRxiv* 2016.

Lu, J.*, et al.* Bracken: estimating species abundance in metagenomics data. *PeerJ Computer Science* 2017;3.

McIntyre, A.B.R.*, et al.* Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. *Genome biology* 2017;18(1).

Mukherjee, S.*, et al.* Large-scale contamination of microbial isolate genomes by Illumina PhiX control. *Stand Genomic Sci* 2015;10:18.

Quince, C.*, et al.* Shotgun metagenomics, from sampling to analysis. *Nat Biotechnol* 2017;35(9):833-844.

Salter, S.J.*, et al.* Reagent and laboratory contamination can critically impact sequence-based microbiome analyses. *BMC Biol.* 2014;12:87.

383 Salzberg, S.L., *et al.* Next-generation sequencing in neuropathologic diagnosis of infections of
384 the nervous system. *Neurology(R) neuroimmunology & neuroinflammation* 2016;3(4):e251.
385 Simner, P.J., Miller, S. and Carroll, K.C. Understanding the Promises and Hurdles of
386 Metagenomic Next-Generation Sequencing as a Diagnostic Tool for Infectious Diseases.
387 *Clinical Infectious Diseases* 2017.
388 Thoendel, M., *et al.* Impact of Contaminating DNA in Whole-Genome Amplification Kits Used
389 for Metagenomic Shotgun Sequencing for Infection Diagnosis. *J Clin Microbiol*
390 2017;55(6):1789-1801.
391 Wood, D.E. and Salzberg, S.L. Kraken: ultrafast metagenomic sequence classification using
392 exact alignments. *Genome Biol* 2014;15(3):R46.
393 Zhang, C., *et al.* Identification of low abundance microbiome in clinical samples using whole
394 genome sequencing. *Genome biology* 2015;16(1).
395