SUPPLEMENTARY INFORMATION FOR KRAKENHLL (BREITWIESER AND SALZBERG, 2018)

1. HyperLogLog algorithm

HyperLogLog is a probabilistic unique count (cardinality) estimator of streams of values with duplicates. It stores a sketch of the data in a concise structure and is very accurate for small cardinalities, keeps constant accuracy rates for up to very high cardinalities.

Intuition. A random bit-string of length *n* can be seen as the outcome of *n* independent binomial trials with $p = 0.5$. Let *k* be the position of the first 1-bit, i.e. the bit string starts with (k-1) 0-bits before the first 1-bit. Since the bits are independent, the probability of *k* is the product of the probabilities, 0.5^k. For example, $k = 6$ means that the bit string starts with 00001_2, and the probability of a random bit-string conforming to the pattern is 0.5^6 or 1/64 (Suppl. Table 1).

k	**Pattern$_k$**	**P$_k$**	**E$_k$**
1	1xxxxxxxxxxx..x	0.5	2
2	01xxxxxxxxxx..x	0.25	4
3	001xxxxxxxxx..x	0.125	8

4	0001xxxxxxx..x	0.0625	16
l	$0^{l-1}1x^{n-l}$	2^{-l}	2^{l}

Supplementary Table 1: The probabilities observing the first 1-bit at position k in a random bit string. E_k shows the expected number of bit-strings we have to observe until seeing one with Pattern_k, and is $1/P_k$.

Inversely, the expected number of independent bit-strings until we see k, E_k is 2^k. If we knew only the maximum number of k, k_{max}, in a stream of independent random bit strings, the best guess at the cardinality of the stream is $2^{k\text{max}}$. Note that this statistic discounts duplicates, as duplicates have the same value. To achieve high precision, HyperLogLog first distributes the stream hashes into $2^p=m$ registers based on the first p bits. The latter 64-p bits are used to determine k_{max} of that register (assuming 64-bit hashes). The final estimate is calculated as harmonic mean of the estimates of all registers. The relative error of the estimate is about $2^{-p/2}$ (see Figure 2).

Algorithm and Implementation. Using 2-bits per base, k-mers up to 31 base pairs can be stored in 64 bits. As k-mers are neither random, nor independently distributed, we hash the k-mers to distribute them uniformly. Good hash functions (a) distribute the input evenly across the output range, and (b) create very different outputs for close inputs (avalanche effect). If both properties are fulfilled for the input (k-mers from different genomes), then we can expect to see precise estimates.

KrakenHLL implements a version of HyperLogLog with the following modifications:

- 64-bit hashes are created by the fast finalizer of the MurMurHash3 algorithm (Appleby, 2017)
- For smaller cardinalities (up to 2^{p-2}) we use a sparse representation that encodes hashes with a much higher precision (Heule, et al., 2013)
- The final estimate is calculated form the register values based on an improved formula (Ertl, 2017)

- The counters can be easily merged for parallel execution. KrakenHLL gives sets of reads to workers, which return HLL sketches in addition to the classification results. The sketches of each taxon are merged into their master sketches by taking the maximum of all register values

Computing the estimate. KrakenHLL implements the recently derived improved estimator for HyperLogLog sketches (Ertl, 2017). Previously proposed methods, including (Flajolet, et al., 2007) and (Heule, et al., 2013), require empirically determined thresholds to account for biases and switching between linear counting and HLL estimator. However, as (Ertl, 2017) shows, the empiric bias correction does not always work.

The raw estimate $\hat{e}_{\text{raw}}$ of Flajolet is based on the harmonic mean of the estimates of the individual registers, times a bias correction factor α_m. Following the notation of (Ertl, 2017), $C:=(C_0, \ldots, C_{q+1})$ is the register histogram, where C_k is the number of registers in M that have the value k.

$$\hat{e}_{\text{raw}} = \frac{\alpha_m m^2}{\sum_{k=0}^{q+1} C_k 2^{-k}}$$

While this works well when the true cardinality λ is in the range $2^p << \lambda << 2^{p+q}$, the estimator is severely biased outside of this range. To account for small range errors, the Flajolet estimator uses linear counting (Whang, et al., 1990) below a threshold of $2.5 \cdot 2^p$:

$$\hat{e}_{\text{small}} = m \log m / C_0$$

While the linear counting estimate is very accurate up to that threshold, the raw estimate that is used above the threshold is still very biased. This can be seen in a big spike in errors in the Flajolet estimate (Suppl. Fig. 1). Heule et al. propose empirically determined bias tables to get rid of the bias. Using observed biases in big amount of random data, they provide correction factors along 200 interpolation points when the raw estimator is in range $\sim 2^p < \hat{e}_{\text{raw}} < 5 \cdot 2^p$. Mostly this correction manages to get rid of the bias (Suppl. Fig. 2), however in some ranges the bias persists (Suppl. Fig. 2).

For large range errors, Flajolet proposes a correction factor when hitting raw estimates above $1/30\, 2^{32}$ (with 32-bit hashes). That factor, however, does not solve the problem but just flips the bias in the opposite direction (Ertl, 2017). When using 64-bit hashes and counting way below 2^{64}, though this bias can be largely ignored (Heule, et al., 2013).

(Ertl, 2017) describes how the biases occur due to not accounting for the fact that the register values are censored at 0 and q+1. Based on the expectation of the censored registers C_0 and C_{q+1}, Ertl derives an improved formula for the estimator without bias:

$$\hat{e}_{\text{ertl}} = \frac{\alpha_\infty m^2}{\sum_{k=1}^{q} C_k 2^{-k} + m\sigma(C_0\, m) + m2^{-q}\tau(1 - C_{q+1}/m)},$$

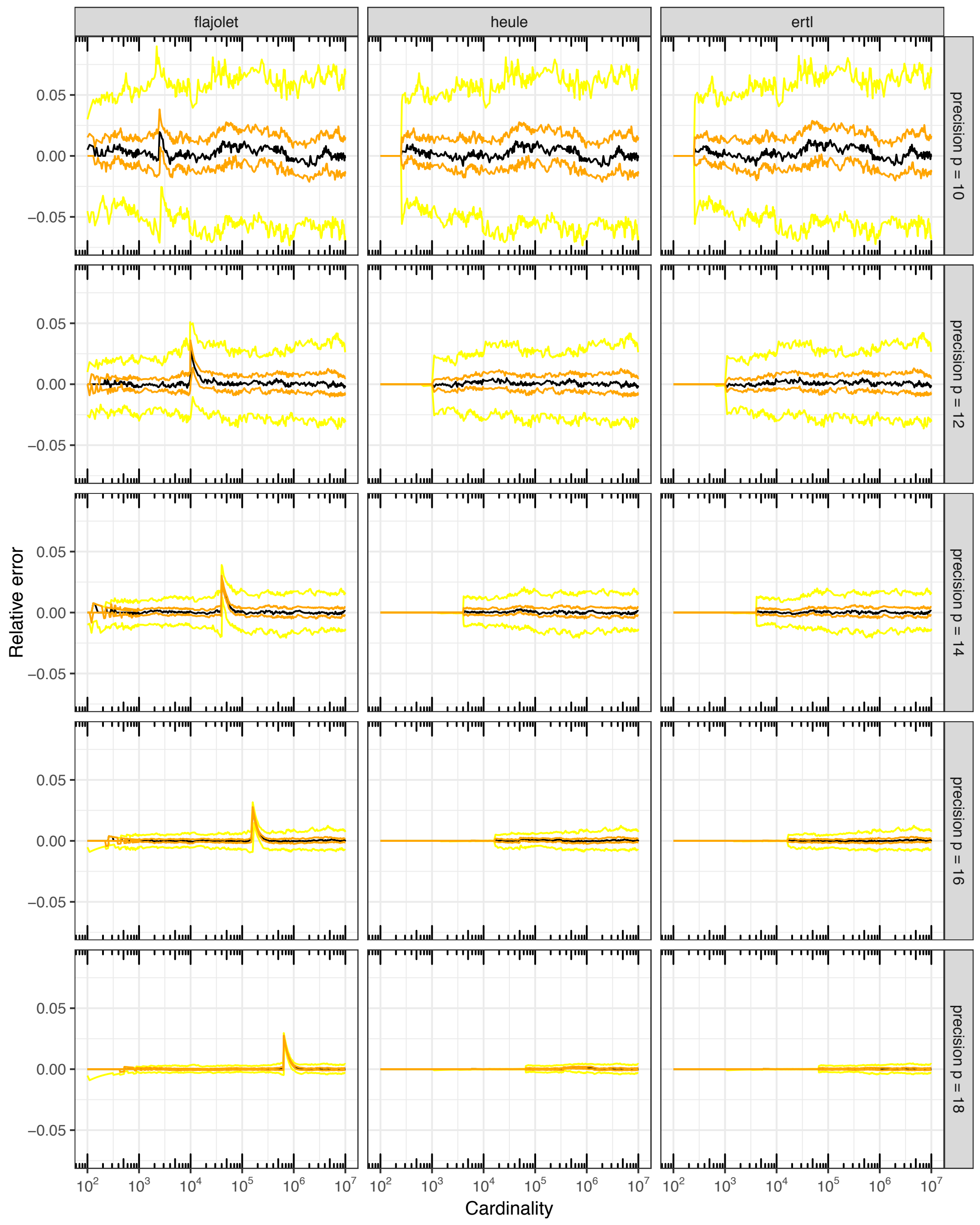
with

$$\sigma(x) := x + \sum_{k=1}^{\infty} x^{2^k}\, 2^{k-1}\, ,$$

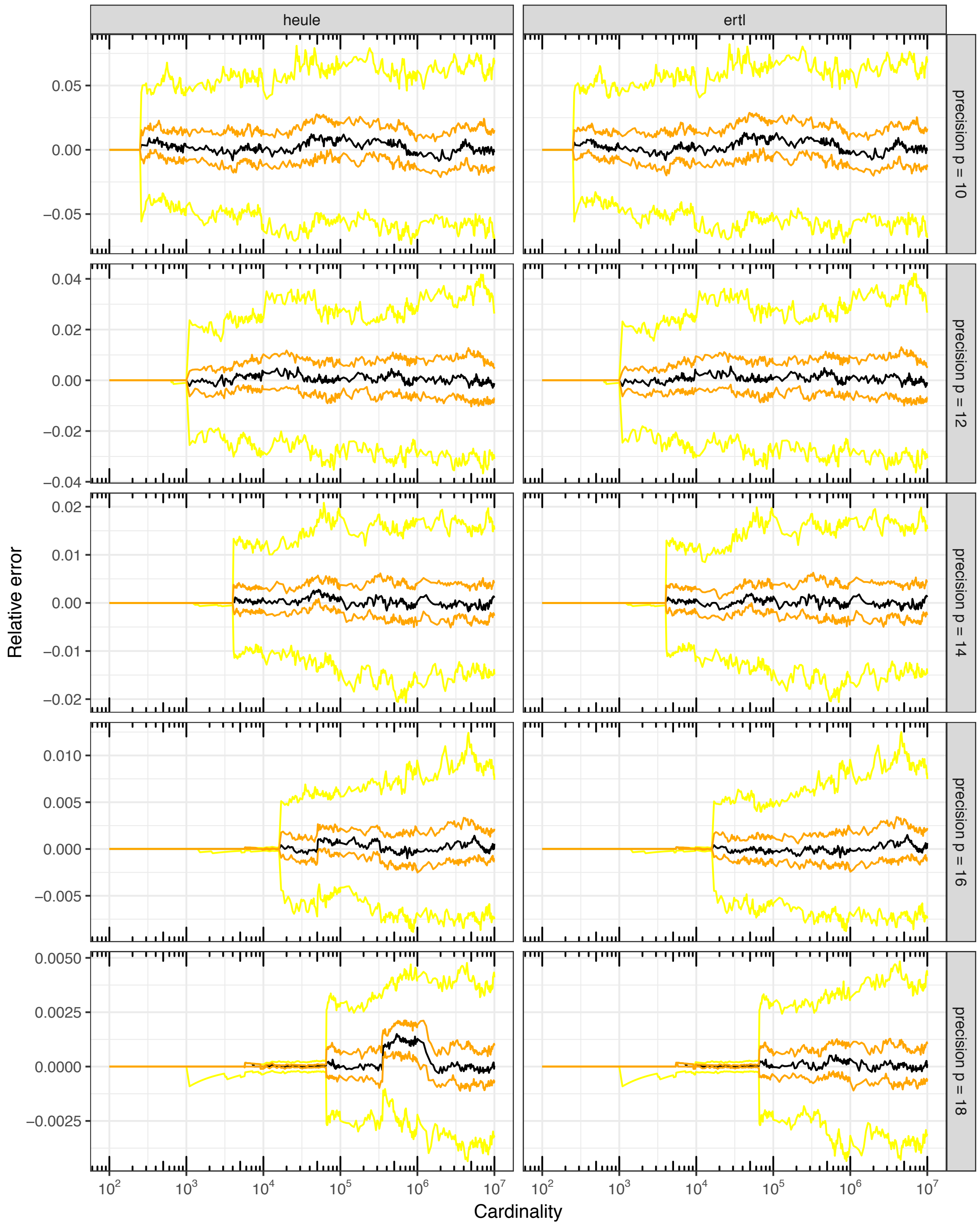
$$\tau(x) := 1 - x - \sum_{k=1}^{\infty} \left(1 - x^{2^{-k}}\right) 2^{-k},$$

$$\alpha_\infty := 1/2 \log 2$$

As seen in Suppl. Figures 1 and 2, the improved estimator of Ertl does not demonstrate any bias. Furthermore, using the sparse representation of Heule et al. for smaller cardinalities gives great precision for lower cardinalities.



Supplementary Figure 1: Comparison of relative errors with Flajolet, Heule and Ertl estimators with varying values of *p*. Black line: median relative error, orange lines 68.2% percentiles, yellow lines 95% percentiles. As expected, the relative error goes down with higher precision values. For both Heule's and Ertl's estimator we use sparse representation for cardinalities up to $2^{p\text{-}2}$ (*p*'= 25). Note that the empirical bias correction of Heule and the mathematical correction of Ertl both get rid of the big spike apparent for Flajolet, when the estimator switches between linear counting and HLL counting. Data from 100 simulated random number runs (64-bit Mersenne Twister seeded with system entropy).



Supplementary Figure 2: Comparison of Heule and Ertl estimators with sparse representation and variable y-axis. At certain precisions and cardinalities, the empirical bias correction values of Heule are not working well. For precision 16, bias is present around cardinalities of 15,000, and for precision 18, bias is present around cardinalities of 150,000 to 1,000,000. Legend: Black line is median relative error, orange lines encompass 68.2% of the estimate errors, yellow lines encompass 95% of the estimate errors. Data from 100 randomly simulated runs of numbers.

2. Database building and reanalysis of patient and test datasets

KrakenHLL includes the new krakenhll-download script to download and dust genomes from specific domains from RefSeq and Genbank. For example, the following command downloads the genomic and rna sequences for all chromosome-level assembled genomes in the category 'vertebrate_mammalian' with taxID 9606 - which gives two human genomes in RefSeq, GRCh38.p11 and CHM1_1.1.1:

```
krakenhll-download --db DB_DIR --fna rna,genomic refseq/vertebrate_mammalian/Chromosome/taxid9606
```

For the reanalysis of the data, we made a database including artificial sequences from UniVec and EmVec, complete viral, archaeal and bacterial genomes, the two human genomes mentioned above, and viral strain sequences (downloaded October 2017). All microbial sequences were dusted:

```
krakenhll-download --db DB_DIR taxonomy contaminants
krakenhll-download --db DB_DIR --dust --include-viral-neighbors refseq/viral/Any
krakenhll-download --db DB_DIR --dust refseq/archaea refseq/bacteria
krakenhll-download --db DB_DIR --fna rna,genomic refseq/vertebrate_mammalian/Chromosome/taxid9606

krakenhll-build --db DB_DIR --build --taxids-for-genomes --taxids-for-sequences --threads 10
```

The database contains 8341 genomes from 3087 prokaryotic species and 139601 sequences from 7295 viral or viroid species. The full database was constructed with 10 threads in 18 hours and is 172GB + 8.1GB for the index. For a performant run a computer with at least 256GB of RAM is required. The samples were run on a machine with four Intel Xeon CPUs E7- 4830 with eight cores each, and 1TB of RAM. The database was built on October 26, 2017. Prior to Kraken and KrakenHLL runs we preloaded the database with krakenhll --preload.

The following command line was used for KrakenHLL:

```
krakenhll --db DB_DIR --threads 10 --report-file SAMPLE.krakenhll.report.tsv --fastq --gzip SAMPLE.fq.gz

> SAMPLE.krakenhll.tsv
```

and Kraken v1.0 was run on the same database with:

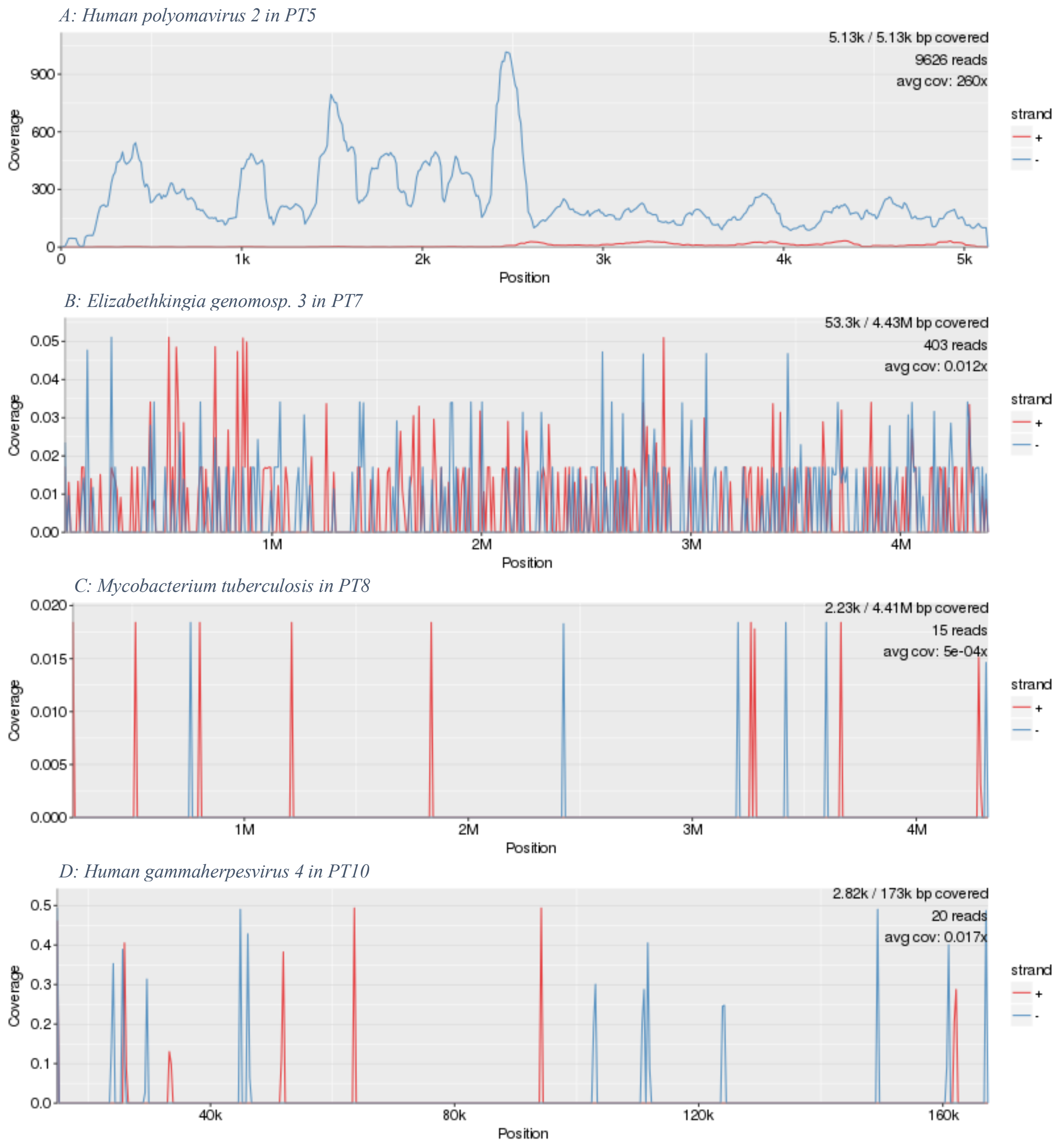
```
kraken --db DB_DIR --threads 10 --fastq --gzip SAMPLE.fq.gz > SAMPLE.kraken.tsv

kraken-report --db DB_DIR --threads 10 --report-file SAMPLE.report.tsv --fastq --gzip SAMPLE.fq.gz >
SAMPLE.kraken.tsv
```

		Speed (Mbp/m)			Wall time (m:s)				Max memory (GB)		
Sample	Number of reads	kraken	krakenhll	speed-up	kraken	kraken-report	krakenhll	speed-up *	kraken	krakenhll	increase
PT1	12022284	487.89	730.99	49.83%	4:35.75	0:55.40	3:11.18	42.27%	122.74	123.07	0.27%
PT2	8294101	483.66	602	24.47%	8:02.34	1:23.18	5:21.21	43.20%	113.16	113.50	0.30%
PT3	17669644	508.34	698.39	37.39%	5:20.30	0:49.66	4:20.68	29.54%	132.56	132.93	0.27%
PT4	29101779	467.33	812.55	73.87%	8:09.75	1:15.19	5:57.75	36.67%	117.43	117.78	0.29%
PT5	26919065	467.43	798.05	70.73%	8:29.84	1:38.30	4:56.05	51.32%	119.61	119.97	0.30%
PT6	27261739	450.36	776.24	72.36%	8:31.67	1:32.81	5:13.75	48.10%	119.68	120.04	0.29%
PT7	19065574	558.09	819.82	46.90%	8:24.83	1:32.52	5:12.86	47.63%	116.45	116.82	0.31%
PT8-S1	6385699	436.51	725.32	66.16%	3:07.28	0:43.75	2:02.68	46.90%	104.54	104.87	0.32%
PT8-S2	7661802	430.89	726.14	68.52%	2:23.71	0:38.51	1:37.75	46.36%	109.25	109.58	0.31%
PT9	26500914	436.63	722.26	65.42%	2:51.90	0:41.73	2:00.31	43.68%	124.42	124.78	0.28%
PT10	21319274	411.94	656.13	59.28%	9:09.45	1:33.31	5:44.53	46.40%	118.19	118.53	0.28%

Supplementary Table 2: Runtime and memory usage for Kraken and KrakenHLL on patient samples (Sazlberg et al., 2016) running with 10 threads. * For kraken, kraken-report was run after classification, and their summed time was compared. KrakenHLL generates the report with the classification binary.

Supplementary Table 3: Test datasets from McIntyre et al. (2017)



Supplementary Figure 4: Coverage of genomes after re-alignment of patient reads. The reads were extracted with krakenhll-extract-reads, aligned against the reference genome with bowtie2(Langmead and Salzberg, 2012), processed with samtools (Li, et al., 2009), and visualized with Pavian (Breitwieser and Salzberg, 2016).

3. Enabling strain and plasmid mappings by storing assembly project and sequence accessions

On the patient samples from (Salzberg, et al., 2016) we how the additional information can also be useful to identify the specific sequences in the database that lead to false positives: In the reanalysis of the samples, *Salmonella enterica* is detected in every sample with up to 233 reads. This species was not detected in the original analysis, and its ubiquity hints that it is a contaminant. However, there are 349 complete genomes in RefSeq for *Salmonella enterica* (taxonomy ID 28901) – and thus in the database, and 23 complete genomes just for *Salmonella enterica subsp. enterica serovar Typhimurium* (taxonomy ID 90371), which is the taxon that is hit most often. Supplementary Table 2 shows a part of the report KrakenHLL generated for PT8. The first indication that it is a false positive identification is the fact that less than 50 k-mers are hit by the 233 reads. Having additional nodes for the assembly and sequence, we find the source of most of the hits: The plasmid pRM9437 of a specific strain. With standard Kraken, neither of this would be known, and additional investigation such as re-alignment of the reads would have been required.

Reads	Taxon Reads	Kmers	TaxID	Rank	Name
233	0	41	590	genus	Salmonella
233	0	41	28901	species	Salmonella enterica
232	0	33	59201	subspecies	Salmonella enterica subsp. enterica
204	0	19	90371	no rank	Salmonella enterica subsp. enterica serovar Typhimurium
203	0	8	1000014850	assembly	GCF_001617585.1 Salmonella enterica subsp. enterica serovar Typhimurium strain=RM9437

203	203	8	1000014852	sequence	NZ_CP014577.1 Salmonella enterica subsp. enterica serovar Typhimurium strain RM9437 plasmid pRM9437, complete sequence

Supplementary Table 3: Part of KrakenHLL output for PT8 of (Salzberg, et al., 2016). Salmonella enterica is a false positive identification, and this is indicated by two factors: (1) the distinct k-mer count is very low. (2) The majority of reads hit just the plasmid of one specific strain. Both of these data are provided by KrakenHLL, but would not be available from a standard Kraken search.

To enable both features, call `krakenhll-build` with the options `--taxids-for-genomes` and `--taxids-for-sequences`. There is important drawback to enabling these options: These pseudo-taxonomy IDs - e.g. 1000014850 in Suppl. Table 2 - are unique to the database build. Special precautions have to be taken when results from different databases are compared or hierarchical matching is used, see next section.

4. Integrating viral strain genomes in the database

The RefSeq project curates viral genomes (Brister, et al., 2015), which are included in the default databases of many metagenomics classifiers. RefSeq includes only one reference genome per viral species, and classifiers that use RefSeq (Kraken and others) therefore only consider those genomes. However, there are thousands of viral strain sequences in GenBank, and the chosen reference genome is often an established but old strain. For example, for HIV-1 the reference is a genome assembly from 1999 (AC GCF_000864765.1), and for JC polyomavirus the reference is the strain Mad1 (AC GCF_000863805.1) assembled in 1993. As viral strain often have higher variability than living organisms, including just the reference genomes in the Kraken database leads to a loss of sensitivity in the detection of strains.

KrakenHLL's database-building script includes the viral strain genomes from the NCBI viral genome resource (Brister, et al., 2015), which maintains a list of 'neighbors' to the viral reference genomes. This list has 112,148 sequences from viral strains, as compared to the 7497 viral genomes in RefSeq (as of October 2017). For example there are over 2500 additional sequences for HIV-1, and over 640 for JC Polyomavirus. In total, these sequences add 100 million (+33%) novel k-mers to the database with k=31. Based on simulated reads from these viral sequences, 21.2% of the reads would not be classified when searching against a database which includes only the RefSeq viral reference genomes.

5. Switching from Kraken to KrakenHLL

KrakenHLL can be used as drop-in replacement to Kraken on a Kraken database. The first run will take longer as KrakenHLL builds its own taxonomy index and counts all k-mers in the database. Note that certain features, such as assembly and sequence identifications, require a full database download and build using KrakenHLL, but the unique k-mer counting works out of the box with a standard Kraken database. Note that `–report-file` on the command line is a required option. To run:

```
krakenhll --db DB --report-file REPORT_FILE --output KRAKEN_FILE
```

6. New taxonomy database format

KrakenHLL has a new taxonomy format based on code from k-SLAM (Ainsworth, et al., 2017). The taxDB file lists the taxa in the following form:

```
Taxonomy ID<tab>Parent Taxonomy ID<tab>Rank<tab>Scientific Name
```

KrakenHLL reports all 27 ranks defined in the NCBI taxonomy, instead of just five abbreviated ranks in Kraken ('D' for superkingdom, 'O' for order, 'P' for phylum, 'F' for family, 'G' for genus, 'S' for species). For example, there are species groups and subgroups, subfamilies and varietas.

7. References for the Supplement

Ainsworth, D., *et al.* k-SLAM: accurate and ultra-fast taxonomic classification and gene identification for large metagenomic data sets. *Nucleic Acids Res.* 2017;45(4):1649-1656.

Breitwieser, F.P. and Salzberg, S.L. Pavian: Interactive analysis of metagenomics data for microbiomics and pathogen identification. *BioRxiv* 2016.

Brister, J.R., *et al.* NCBI viral genomes resource. *Nucleic Acids Res* 2015;43(Database issue):D571-577.

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.

Langmead, B. and Salzberg, S.L. Fast gapped-read alignment with Bowtie 2. *Nat Methods* 2012;9(4):357-359.

Li, H.*, et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 2009;25(16):2078-2079.

Salzberg, S.L.*, et al.* Next-generation sequencing in neuropathologic diagnosis of infections of the nervous system. *Neurology(R) neuroimmunology & neuroinflammation* 2016;3(4):e251.

Whang, K.-Y., Vander-Zanden, B.T. and Taylor, H.M. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Syst.* 1990;15(2):208-229.