Haploflow: Strain-resolved *de novo* assembly of viral genomes

A. Fritz[1,2], A. Bremges[1,2], Z.-L. Deng[1], T.-R. Lesker[1], J. Götting[2,3], T. Ganzenmüller[2,3,4], A. Sczyrba[1,5], A. Dilthey[6,7], F. Klawonn[8, 9], A.C. McHardy[1,2]*

[1] BIFO, Department of Computational Biology, Helmholtz Centre for Infection Research, Braunschweig, Germany

[2] DZIF, German Centre for Infection Research

[3] Institute of Virology, Hannover Medical School, Hannover, Germany

[4] Institute for Medical Virology, University Hospital Tuebingen, Tuebingen, Germany

[5] Faculty of Technology and Center for Biotechnology, Bielefeld University, Bielefeld, Germany

[6] Institute of Medical Microbiology and Hospital Hygiene, University Hospital, Heinrich-Heine-University Düsseldorf, Düsseldorf, Germany

[7] Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, Bethesda, MD, 20892, USA

[8] Department of Computer Science, Ostfalia University of Applied Sciences, Wolfenbuettel, Germany

[9] Biostatistics Group, Helmholtz Centre for Infection Research, Braunschweig, Germany

*send correspondence to: alice.mchardy@helmholtz-hzi.de

In viral infections often multiple related viral strains are present, due to coinfection or within-host evolution. We describe Haploflow, a de Bruijn graph-based assembler for *de novo* genome assembly of viral strains from mixed sequence samples using a novel flow algorithm. We assessed Haploflow across multiple benchmark data sets of increasing complexity, showing that Haploflow is faster and more accurate than viral haplotype assemblers and generic metagenome assemblers not aiming to reconstruct strains. Haplotype reconstructed high-quality strain-resolved assemblies from clinical HCMV samples and SARS-CoV-2 genomes from wastewater metagenomes identical to genomes from clinical isolates.

Due to co-infection or within host evolution, in viral infections closely related strains, or haplotypes, might be present, with high average nucleotide identity (ANI)[1] to one another[2–5]. Modern sequencing technologies can capture this variation and computational assembly techniques reconstruct the individual genomes from the resulting data. Currently there are predominantly two types of methods for this problem, viral haplotype assemblers[6,7] and general (meta)genome assemblers[8–12]. Assembly of individual strains is very difficult, especially if variation is low and few reads span varying sites, resulting in highly fragmented strain genome reconstructions or consensus assemblies[13,14].

(Meta)genome assemblers usually represent read data initially as a deBruijn (kmer) graph and haplotype assemblers use string graphs[15–18]. String graphs, while being computationally more expensive to construct[19], due to overlap calculation for all read pairs, have the advantage of detecting mutations that co-occur on a single read[6], while for deBruijn graphs, this is limited to mutations occurring within the specified *k*-mer length[20]. String graphs are thus more sensitive in matching mutations to strains. If the strains have long stretches of identical sequences, co-occurrences may not happen, which typically is then solved by returning fragmented genome assemblies, where contigs are split between consecutive mutations that cannot be assigned to individual strains. As more contextual information is lost in the deBruijn graph, mutations appear as “bubbles” in the graph, where consecutive vertices are connected by more than one edge[21,22]. (Meta)genome assemblers typically consider these bubbles as errors and follow different approaches for their resolution[22]. The popular SPAdes assembler only considers one path of the bubble, and thus loses the information of the second strain and reconstructs the dominant strain[9,13,23]. MEGAHIT instead terminates contigs prematurely if a bubble is encountered[8]. This leads to fragmented assemblies in the presence of closely related strains[13].

We here describe Haploflow, a new method and software for the de novo, strain-resolved assembly of viral genomes, which overcomes the problems for both types of methods, i.e. low speed versus loss of strain-specific information, by using information on differential coverage between strains to deconvolute the assembly graph into strain resolved genome assemblies. Haploflow thus does not require reads spanning multiple variable sites for strain resolved assembly of low divergent haplotype populations. As it is based on deBruijn graphs it approaches the runtime behaviour of modern metagenome assemblers. We demonstrate the

ability of Haploflow to resolve strains fast and accurately on multiple data sets, including a low complexity HIV strain mixture to a complex, simulated virome sample consisting of 572 viruses with substantial strain-level variation, varying abundances and genome sizes as well as two data sets of clinical human cytomegalovirus (HCMV) and SARS-CoV-2 data.

Results

We next describe the algorithm for creating and manipulating the assembly graph and the flow algorithm that gave Haploflow its name.

deBruijn and unitig graph creation

The input to Haploflow is a sequence file including read sequences and specifying the k-mer size for constructing the deBruijn graph. Optionally, the lowest expected strain abundance (or *error rate*) can be specified, leading to removal of more rare kmers from the graph, for graph simplification. Setting the *error-rate* size too low possibly makes the unitig graph and subsequent assembly more complex, while a too high value will prevent low abundant strains from being assembled.

First, a deBruijn graph[21] is created from the reads, using ntHash[24] for kmer hashing. Given the reads $R = \{r_1, ..., r_n\}$, the deBruijn graph $G = (V, E, k)$ contains all substrings of length k of R as vertices V and two vertices u and v are connected with an edge if the prefix of u overlaps with the suffix of length $k-1$ of v or vice versa[25], i.e. $(u, v) \in E \Leftrightarrow u_{1...k-1} = v_{2...k} \vee u_{2...k} = v_{1...k-1}$. In addition, k-mer counts for every encountered kmer are stored and all weakly connected components (called CCs, a set of vertices that are connected directly or indirectly to each other in the graph) of the graph are calculated. The connected components are found with repeated depth-first searches, until every vertex has been visited and its connected component set. Afterwards, CCs are transformed individually into condensed versions of deBruijn graphs, so-called unitig graphs, where linear paths of vertices, having only one ingoing and one outgoing edge, are collapsed into one vertex.

This unitig graph has the following properties:

a) Every remaining vertex is a junction, having more than one ingoing or outgoing edge or being a source or sink. This means that all variation is found in vertices, all non-unique sequences (i.e. occurring in multiple haplotypes) are found in edges.
b) The unitig graph is a homeomorphic image of the input deBruijn graph, disregarding error correction. This means that no information is lost and the original deBruijn graph could be reconstructed.

When constructing this unitig graph, for each connected component, so-called junctions, vertices having a different in- from out-degree, or an in- or out-degree of more than one in the de Bruijn graph are identified with a depth-first search. These will be the vertices of the new unitig graph, and their kmers are maintained (Supplementary Fig. S1). The sequence of all the traversed kmers is added to the connecting edge and we define the length of an edge as the length of this sequence in base pairs. Starting from any junction, the next junction in the deBruijn graph is searched, passing vertices with exactly one ingoing and one outgoing edge until the next junction is found. Since all junctions are guaranteed to be searched and the transformation is deterministic, the choice of starting junction does not matter. When the next junction is found, the coverage of all the traversed edges is averaged and checked versus a threshold based on the *error rate* (Supplementary Fig. S2). If it is above, the target junction is also added as a vertex to the unitig graph and an edge with the (averaged) coverage value as the edges coverage is added between the two vertices. If the coverage is below the threshold, then neither the target vertex nor the edge are created and the next outgoing edge of the source is considered. This is repeated until all junctions have been searched, such that no vertices with in-degree = out-degree = 1 are remaining (Figure S1). The resulting unitig graph is usually of drastically reduced size in comparison to the original graph, with sometimes less than 0.01% of vertices remaining. All linear paths of the original graph are condensed into single edges that represent stretches of unique contig sequences.

For every unitig graph a kmer coverage histogram is built (Fig. S2). These histograms reveal several key properties on our data sets: First, the coverage of reads belonging to one genome is approximately normally distributed around the "real" coverage of that genome[19,20] If multiple sufficiently distinct (in terms of average nucleotide identity) genomes are present in a single unitig graph, then all of them will have a corresponding peak in the histogram. The

longer a genome, the more different kmers it includes, and accordingly, the higher the peak. If genomes are very closely related, then the peaks will consist of *k*-mers that are unique to the individual strains and there will be another peak for the common *k*-mers.

Haploflow uses these coverage histograms as indication of the putative number of genomes[26] and their size relation as well as for error correction. Every read error will create *k* erroneous kmer vertices in the deBruijn graph[27,22], with low coverage in comparison to the real coverage *cov* of the genomes. Since sequencing errors are rare in Illumina reads, most erroneous kmers will only appear once[28,29], with fewer kmers appearing multiple times, creating an exponentially decreasing curve in the kmer histogram. This information is factored into the error correction with too rare *k*-mers being removed (red line, Figure S2). The exact method and values used for error correction can be customized by the user, but by default, all *k*-mers with a coverage less than the first inflection point of the coverage histogram are filtered and every *k*-mer which has less than 2% of the coverage of its neighbouring *k*-mers. This parameter can be increased when dealing with long read data to reflect the higher number of errors in current long read technologies.

Assembly using the flow algorithm

In the second stage the algorithm operates on the unitig graph. It infers and returns a set of contigs based on paths of similar coverages throughout the graph. The flow algorithm consists of three steps that are repeated until the whole graph has been resolved into contigs: (i) finding paths through the graph, (ii) assigning flow values to them, and (iii) determining the path sequence.

In the first step, the source vertex (with an in-degree of 0) with the highest coverage is selected from the unitig graph. Starting from this source, a modified Dijkstra's algorithm[30] is applied, which identifies the fattest path from a source to sink (a vertex having an out-degree of 0) based on edge coverages (Alg. 1, Fig. 1). The fatness of a path is defined by the minimal fatness of the edges on the path. The fatness of an edge is determined as the minimum of its coverage and the fatness of the path from the source until the current edge[31] and can also be called the "capacity" of the edge. The fattest path from a source to a sink is

then determined by following the edges maximising fatness until the sink is found. All edges on this path are then marked with a path number. Subsequently, the coverage for all edges on this path are reduced by the path fatness, the next source is selected and the previous steps are repeated until no edges with coverage remain.

Likely due to technical issues, such as amplification biases[32] and read errors[33], and biological structures such as genomic repeats[34], coverages do not follow a normal distribution globally and consequently some consecutive edges in the assembly graph may exhibit steep changes in coverage. This is the reason why Haploflow uses a two-step procedure for path finding: First, paths are found through the graph as described before. But instead of directly returning contigs for these paths, these paths are only putative, meaning that all paths and changes to the graph are temporary first.

The algorithm of Haploflow is then able to handle heterogeneous coverages across genomes, e.g. highly pronounced in amplicon data or sequence data with high error rates, by using the local, not global coverage distribution, and not absolute coverage, but relative coverage, i.e. the only assumption is that the ratio between haplotypes is somewhat conserved. Additionally, putative paths can get removed, if too many of its edges are already part of a previous putative path (Supplementary Methods). If a path consists almost only of edges that have been used before, it is an indicator that these paths would lead to duplicated contigs. Finally, this results in a graph where all edges are marked with one or more paths they are assumed to be on.

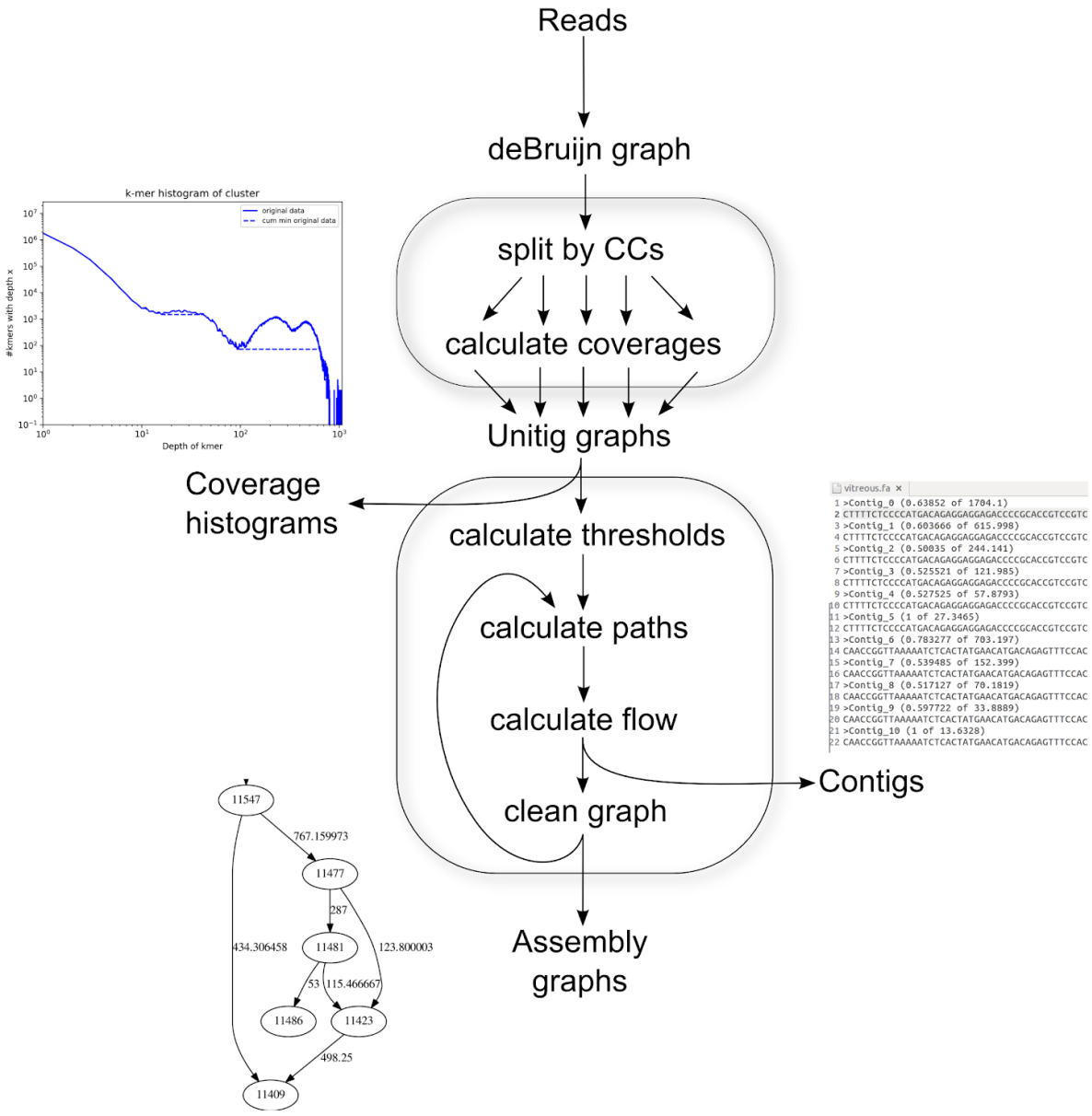


Figure 1: Flow chart of the Haploflow algorithm and its two parts: First the construction of the deBruijn graph and operations thereon, namely splitting it by connected components and calculating coverages. Then the creation of the unitig graphs per CC and the assembly process consisting of calculating the thresholds and the coverage histograms and the putative paths through the graphs. Next is the calculation of the concrete flows and thereby the generation of the contigs and finally the cleaning of the graph and the generation of the assembly graphs. As intermediate output the assembly graph is created during every step (bottom left).

Algorithm 1: Fattest-path Dijkstra

Data: graph $G = (V, E)$ with edges $e = (v, w) \in E$, with $v, w \in V$, source $s \in V$, non-negative edge capacities $c(v, w)$ with $v, w \in V$
Result: Graph with edges labelled by their fatness, starting from source $s \in V$

1 **forall the** $v \in V - \{s\}$ **do**
2 $\quad v.fat = 0$ set fatness to 0
3 $\quad s.dist = \infty$ set distance to ∞
4 **end**
5 $Q = priority\_queue(V)$ create priority queue keyed by $v.dist$
6 **while** $!(Q.isEmpty())$ **do**
7 $\quad u = argmax(Q, fat)$ select fattest vertex u in Q
8 $\quad del(u, Q)$ remove u from Q
9 $\quad$ **forall the** $v \in V$ *s.t.* $(u, v) \in E$ **do**
10 $\quad\quad$ Breadth-first search through G
11 $\quad\quad$ **if** $v.fat < min\{u.fat, c(u, v)\}$ **then**
12 $\quad\quad\quad v.fat = min\{u.fat, c(u, v)\}$ update fatness
13 $\quad\quad\quad v.dist = u.dist + length(e)$ set $dist$ as distance to source
14 $\quad\quad\quad$ update Q with new $dist$ values update Q
15 $\quad\quad\quad u.pred = v$ set predecessor for backtracking
16 $\quad\quad$ **end**
17 $\quad$ **end**
18 **end**

Alg. 1: The adapted Dijkstra algorithm used in Haploflow to find fattest paths through the unitig graph. Instead of determining the shortest paths from the source to all vertices, this algorithm determines the fattest path. The fatness is initialised as 0 for all vertices but the source and then the graph is searched using a breadth-first search and based on the fact that the fattest path from a source *s* to a sink *t* is based on the edge with the lowest coverage along this path (lines 9 to 12).

In the second part of the path finding we start again from the source with the highest coverage. Since we have all edges marked with the path that they are on, we can select the edge on the same path which is farthest away from our source and calculate the fattest path from the source to this sink. If Haploflow is not able to resolve the fatness unambiguously,

for example because two outgoing edges have almost the same fatness, then the path is terminated in this vertex. This is to prevent formation of chimeric contigs, because locally two strains might have similar coverages. For the final path, a corresponding contig is returned and the coverage reduced permanently (Supplementary Methods). Then all edges with capacity 0 and all vertices without any edges are removed and the flow algorithm started anew from the source vertex. This procedure is repeated until the graph does not have any edges remaining.

Haploflow has multiple parameters that can be set to improve the assembly, if more information is given. If no additional information is given, Haploflow has default settings that usually already provide high quality assemblies. All the evaluations in this article were performed using these default parameters, i.e. a value for k of 41, and an *error-rate* of 0.02. The value of $k = 41$ was chosen since too small (in comparison to read lengths) values for k lead to more ambiguities and a higher k might lead to fragmented assemblies. If k does not exceed 50% of read-size, the assemblies are of comparable quality. The error-rate parameter was set to 0.02, because this is the value assumed to be the upper bound of errors in short-read sequencing[35] and can be increased when dealing with more error-prone reads like those from PacBio or Oxford Nanopore.

Additional parameters include a setting for detecting strains with very low absolute abundance (*strict*), for data sets with exactly two strains (*two-strain*), as well as an experimental mode for highly complex data sets with clusters containing five or more closely related strains.

SARS-CoV-2 clinical and wastewater metagenome data

We reconstructed viral haplotypes using Haploflow from 17 clinical SARS-CoV-2 samples sampled in Northrhine-Westphalia, Germany (DUS, 5 Illumina short-read samples) and Madison, Wisconsin (WIS, 6 Illumina short-read and 6 Oxford Nanopore long-read samples). After correcting for PCR amplification and sequencing errors (Supplementary Methods), Haploflow identified two strains in nine samples, consistent with in-sample variation[36–38]. The assembled contigs were assessed with QUAST[39] using the Wuhan-Hu-1 isolate strain (RefSeq NC_045512.2) as reference genome. For all samples, Haploflow produced contigs spanning the complete genome, in 13 cases as a single contig. Haploflow reconstructed the consensus genome sequence(s) found in GISAID[40] with almost 100% identities as the major strain -

from both Illumina and MinION data generated for all WIS samples (Fig. 2). For the Wisconsin strains, which were passaged for up to two rounds in cell cultures, the reconstructed minor strains from short read data had more evolutionary divergences. In comparison to calls from the variant caller Lofreq[41], (Supplementary Methods), which performs particularly well on mixed strain viral data[14], both identified 17 (65.4%) of overall 26 identified variant sites (mutations and up to 2bp indels). Interestingly, most of these are C->T transitions, indicating a tendency to alter genome composition[42] (Supplementary Table S6). In addition, Haploflow identified three longer deletions. Five (19.2%) "unique" LoFreq variants are located in error-prone regions (homopolymeric or strand biased) or at the very end of the genome. Four further low frequency sites (<5%, 15.4%) were found by Haploflow and were also among low frequency Lofreq predictions.

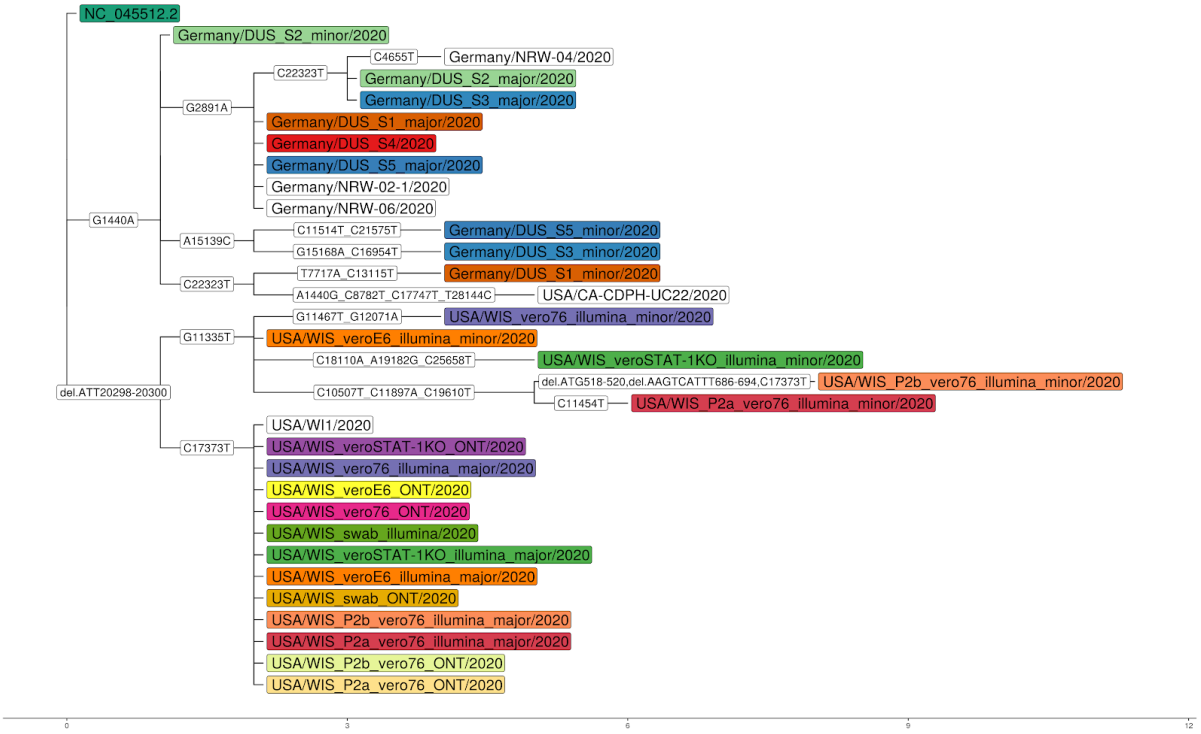


Figure 2: Phylogenetic relationships of reconstructed strain genomes inferred with Raxml[54,55], including closely related (ANI greater than 99.99%, determined with MASH[56]) strains from GISAID[57]. Strains from the same sample are indicated by color, and "major" and "minor", based on their inferred abundances. Evolutionary events, including mutations and indels are shown on edges.

In a study of eight shotgun metagenome samples of sewage from the San Francisco Bay Area[43], the authors manually assembled consensus SARS-CoV-2 genomes from seven samples and subsequently called variants with inStrain[44]. A comparison to common variants of clinical isolate genomes showed that most of the SNPs found in the data set could be detected in the isolate genomes, with the more (>10%) abundant ones found in strains from California or the US. This and the abundance distribution of some SNPs over time suggested that the data set captured real genomic variation and that different SARS-CoV-2 strains were present in this data set. Haploflow with the option *strict 1* (reduced error correction threshold to account for shallow sequencing depth) and scaffolding (Supplement), assembled full-length SARS-CoV-2 genomes for the same seven samples, recovering two strains for six of them (Supplementary Table S8). Strikingly, for all assemblies identical genomes of clinical SARS-CoV-2 isolates were identified in the GISAID database using minimap[45] v2.17 (Supplementary Table S8), mostly from samples obtained in the U.S. (5), and California (3), highlighting the ability of Haploflow to recover high quality, strain-resolved viral haplotype genomes from metagenomic data.

Performance evaluation

We evaluated Haploflow on three simulated data sets with increasing complexity: a mixture of three HIV strains represented by error-free simulated reads, multiple in-vitro created mixtures with different proportions of two HCMV strains sequenced with Illumina HiSeq[46], and a simulated virome[47,48] data set of 572 viruses, with 417 genomes in unique taxa and 155 genomes in common strain taxa with up to eleven closely related strains, to assess Haploflow's ability to assemble complex, larger data sets. Finally, we assembled HCMV genome data from clinical samples collected longitudinally over time from different patients[49], to characterize the within- and across patient genomic diversity of viral strains, including also larger genomic differences between individual strains in mixed-strain infections, which has not been possible so far. The evaluation was performed using metaQUAST[50] v.5.0.2, which is commonly used to evaluate metagenome assemblies and provides useful metrics for measuring completeness (genome fraction), continuity (NGA50, largest alignment) and accuracy (mismatches per 100kb, duplication ratio) of assemblies and has specific options for analyzing strain-resolved assemblies. In addition, we calculated

metrics for assessing strain-resolved assembly; the strain recall, specifying the fraction of correctly assembled strains (more than 90 (80)% genome fraction and less than 1 (5) mismatches/kb), the strain precision, specifying the fraction of correctly assembled strain genomes of all provided genome assemblies (true positives defined as in recall; total number of genome assemblies estimated as number of ground truth genomes with at least one mapping contig * duplication ratio), as well as the composite assembly quality score, we previously defined[14]. This composite score takes six common assembly metrics (genome fraction, largest alignment, duplication ratio, mismatches per 100 kb, number of contigs and NGA50), normalises them in the range of all results, such that $score(method) = \frac{value(method) - min(value(m \in methods))}{max(value(m \in methods)) - min(value(m \in methods))}$ for genome fraction, largest alignment and NGA50 and $score(method) = \frac{max(value(m \in methods)) - value(method)}{max(value(m \in methods)) - min(value(m \in methods))}$ for the other metrics and then weighs with a weight of 0.3 for genome fraction and largest alignment, respectively and a weight of 0.1 for the other metrics.

HIV-3 in silico mixture

HIV, the human immunodeficiency virus, is a single-stranded RNA virus with an approximately 9.5 kb genome that infects humans, causing AIDS (acquired immunodeficiency syndrome). HIV evolves rapidly within the host and may also present as multi-strain infections[51,52]. The three HIV-1 strains 89.6, HXB2 and JR-CSF, which are commonly used to evaluate viral haplotype assemblers[53,54], were downloaded from NCBI RefSeq[55], mixed in the proportions 10:5:2 and error-free reads with a length of 150bp and depth of 20,000 created with CAMISIM[56] and the wgsim read simulator[57]. These genomes differ mainly by SNPs and have an average nucleotide identity (ANI) of ~95%. This threshold was chosen, because experiments on MEGAHIT and metaSPAdes showed that genomes more closely related than 95% will not be resolved[56].

We benchmarked the quality of strain-resolved Haploflow assemblies for the three strain HIV data against five other *de novo* assemblers (SPAdes, metaSPAdes, megahit, PEHaplo, SAVAGE in *de novo* mode) with metaQUAST v.5.0.2, using multiple parameter settings, if defaults settings were undefined (QuasiRecomb, PEHaplo). Furthermore, we assessed five reference-based assemblers (GAEseq[58], SAVAGE ref-based mode, PredictHaplo, QuasiRecomb and CliqueSNV), which were provided with one strain genome for assembly.

Of all evaluated *de novo* assemblers, Haploflow **performed best across all metrics and the composite assembly score (**Figure S2**)**, assembling all three strains almost completely (more than 90%), with less than 1 mismatch/kb, providing no false positive strain assemblies - that for some methods (QuasiRecomb) reached several thousand strains - and with more than double the assembly contiguity (NGA50) than the second best method (PEHaplo). Haploflow was the only method assembling all strain genomes into complete contigs. Also in comparison to the reference-based assemblers, Haploflow performed best. SAVAGE in reference-based mode, run on a subsample of the data, performed similarly well in five of the eight metrics, however, provided a substantially more fragmented assembly (lower NGA50, more contigs) and a strain genome with more mismatches. Haploflow also closely estimated the true underlying strain proportions, with predicted coverages of 10,371 for HIV 89.6, 5,372 for HIV HXB2 and 1,745 for HIV JR-CSF.

HCMV in vitro mixtures

We next evaluated Haploflow on six lab-created mixtures of two HCMV strains sequenced with Illumina MiSeq[59]. HCMV is one of the largest human pathogenic viruses, causing severe illness in immunocompromised patients and infants, and possessing a double stranded DNA genome of more than 220 kb[60]. The data set includes two different strain mixtures, denoted "TA" (strains TB40 and AD169, 97.9% ANI) and "TM" (strains TB40 and Merlin, 97.7% ANI), with three different mixture ratios each (1:1, 1:10 and 1:50), allowing us to test the ability of assemblers to resolve strains at varying abundances. We ran Haploflow on these data and compared the results to those of twelve other assemblers. These include nine (meta)genome assemblers (ABySS, IDBA, MEGAHIT, metaSPAdes, Ray, SPAdes, tadpole, IVA[61] and Vicuna[62]) also widely used for single-cell and virome data because of their accuracies and speed, and three specialised viral haplotype assemblers delivering a result (reference-based SAVAGE, VirGenA[63] and PEHaplo). Four more viral haplotype assemblers, SAVAGE *de novo*, PredictHaplo[64], CliqueSNV[65] QuasiRecomb[46], either did not return results[14] or were terminated after ten days (Haploflow on average required 1.5h per sample). Assemblies were evaluated using metaQUAST v.5.0.2 with the benchmarking workflow QuasiModo[14], based on common assembly metrics, the composite assembly score, recall and

precision in strain-resolved genome assembly, as before, and the top performing methods falling in the 95-100% range of results identified for every metric.

Of the 12 evaluated de novo assemblers, Haploflow scored best in 5 of the 8 metrics, followed by metaSPAdes (best in 2 of 8: NGA50, duplication ratio), while PEHaplo, tadpole, IDBA, Vicuna and IVA each scored best for one metric, respectively (Supplementary Table S2). Haploflow assemblies were of very high quality, recovering the most correct strain genomes (10 of 12), providing the best strain precision and composite assembly score (9.34 of 10), highest genome fraction (83.87%) and the most contiguous assemblies (NGA50 62,560). Interestingly, the similarly good NGA50 values of metaSPAdes and Haploflow were obtained in different ways, for the former due to a more contiguous assembly for the abundant strain, while only Haploflow and the haplotype assembler SAVAGE in reference-mode recovered more than 50% of the low abundant strain in several mixtures.

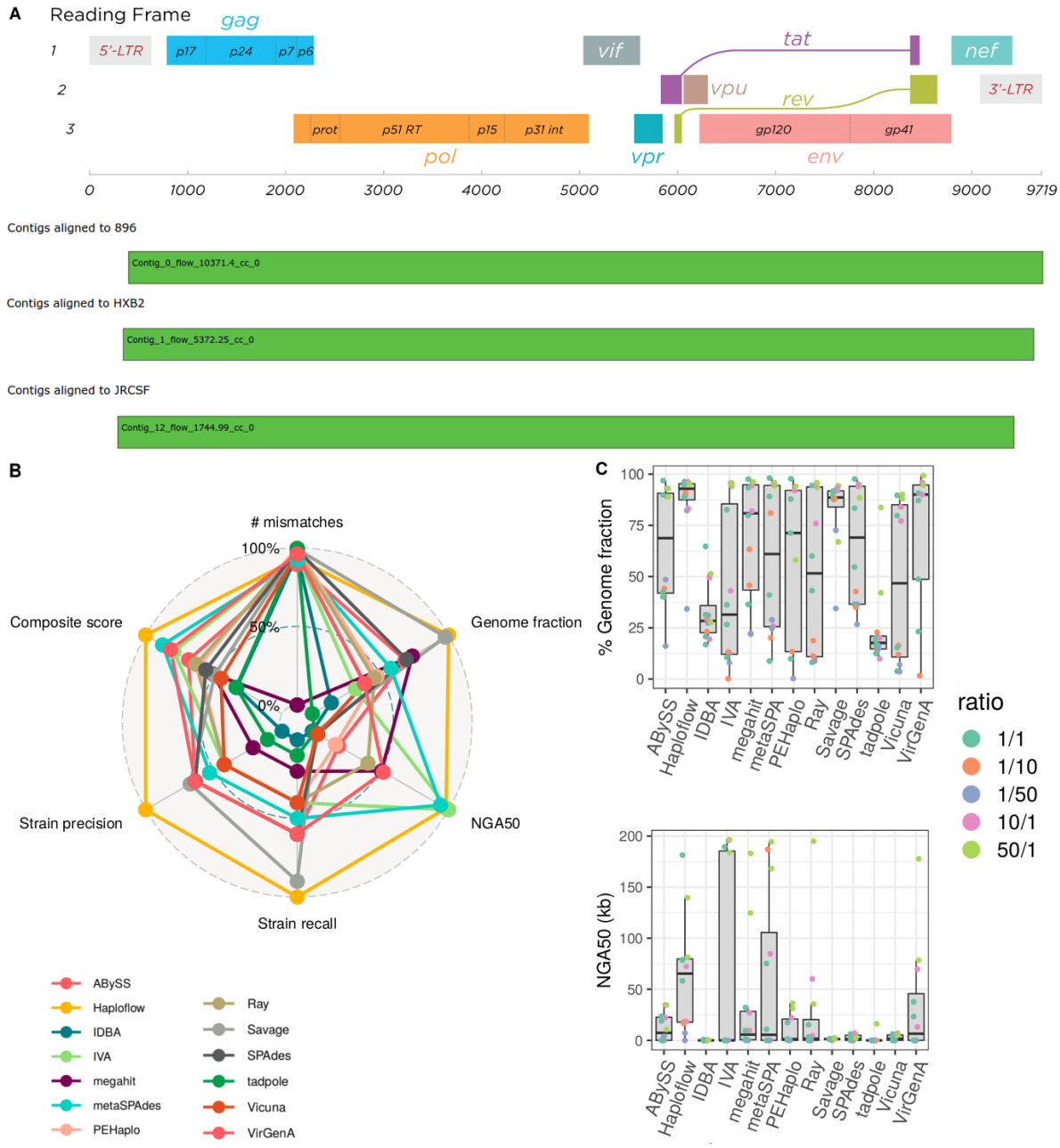


Figure 3: **A:** HI virus genome structure[66] and Icarus plots[67] for three HIV strains reconstructed by Haploflow. For each of the three reference genomes there is one contig spanning almost the complete genome. **B:** Radar plot of relative performance with commonly used and strain-resolved genome assembly metrics for Haploflow and 12 other methods on the HCMV benchmark data (best values are at 100%, see Performance evaluation). Haploflow, in orange, ranks first in genome fraction, Strain recall, Strain precision and Composite score. **C:** Boxplots with median and interquartile range of genome fraction and NGA50 values across samples for different methods.

Simulated virome data set

To test Haploflows ability to recover viral strain genomes from complex data sets, we evaluated Haploflow, MEGAHIT and metaSPAdes on the simulated virome data set from the Namib desert[47], which includes short-read data simulated from an *in-silico* mixture of 572 viral genomes created to assess different assemblers[48]. It was not possible to run the *reference-free* haplotype-assemblers (SAVAGE, PEHaplo) on this data set. To assess the evolutionary divergence between the viral genomes, we identified clusters of similar genomes using dRep[68], which resulted in 469 clusters total, out of which 52 clusters had at least two members with more than 95% ANI (average nucleotide identity), resulting in 417 "unique" genomes and 155 genomes in common strain clusters. The 95% threshold was chosen since MEGAHIT and metaSPAdes are only able to resolve genomes less similar than that[56].

For the 155 common strain genomes, Haploflow correctly assembled 13-28.6% more sequence (62.85% genome fraction versus 55.58% and 48.88% for SPAdes and MEGAHIT, respectively). This was even more pronounced for clusters with genomes of at least eight-fold coverage, for which 19.8-37.5% more genome sequence was correctly assembled (89.37% versus 74.58% and 64.99% for SPAdes and MEGAHIT, respectively). For the less abundant strains from these clusters, 32.7-45.3% more genome sequence was correctly assembled (87.37% versus 65.85% and 60.12% genome fraction, respectively). Even for the complete data set with "unique" genomes and low abundant genomes, Haploflow reconstructed genome fractions similar to the MEGAHIT and metaSPAdes assemblers (72.2% and 68.6% versus 66.6% genome fraction; Table S5), which performed best in the original publication.

Analysis of clinical HCMV data

We used Haploflow with default parameters to reconstruct genomes from longitudinal clinical samples of eight HCMV positive patients, who had multi-strain infections[59] (Supplementary Table S5). QUAST was used to map HaploFlow's contigs against the consensus strain of the first time point as reference genome, as the exact underlying strain genomes in the samples are unknown. Using the QUAST output, in particular the duplication ratio, the number of strains predicted by HaploFlow was determined by rounding the duplication ratio and then clustering the contigs into that many clusters using HaploFlow's predicted flow (using

python's sklearn[69] *k*-means method). For each of the clusters, QUAST was re-run, again using the consensus as reference genome. Since the resulting genomes, in particular the low abundant (minor) strains, will inherently be different to the consensus to some degree, only the genome fraction is considered a relevant metric here. Additionally, to confirm that HaploFlow created accurate strain-resolved contigs instead of consensus contigs, we compared clusters from the same patient at different time points with each other, finding that contigs from two clusters from consecutive time points showed ~99.9% ANI, while randomly matched clusters only had ~98% ANI.

For all patients with multi-strain infections, Haploflow reconstructed multiple complete genomes for at least one time point. For most patients, sequencing data of their infection exist for multiple time points and Haploflow recovered all strains, if the abundance of the lower abundant strain exceeded 6.8%, once as low as 6.1% (Supplementary Table S5). Haploflow reconstructed the genomes of both, or in two cases three, strains infecting the patient. Haploflow correctly predicted at least one lower abundant strain for 19 of 23 (82.6%) time points with multiple strain infections, correctly predicted 44 strains of the total 48 strains (91.7% recall) and only predicted (parts of) three unconfirmed, additional strains (93.2% precision). Haploflow also reproduced the results from the original publication[59], where a strain with a structurally altered genome established itself as dominant over two consecutive time points in patients SCTR1 and SCTR11, and also recovered three distinct strains from the SCTR18 sample (Supplementary Table S5). The samples for which Haploflow did not assemble a second strain had either a very low abundant second strain (4% for SCTR1-day91), a shallow coverage (coverage of 22 and 38 for SCTR1-245days and SCTR3-320days) or a combination therof (6.8% variant at 105 coverage for SCTR1-194days).

Finally, we tested Haploflow on a HCMV sample for which the genotypes and proportion of both strains were known (Supplementary Figure 4). Haploflow reconstructed both strains with a total of 19 contigs, 4 for the high abundant strain and 15 for the low abundant. The high abundant strain assembly matched the consensus strain with 0.87 mismatches/100kb, a NGA50 value of 113,718 and 99.84% genome fraction. The contigs produced for the low abundant strain were also evaluated using the consensus sequence, showing 94.58% genome

fraction and an NGA50 of 62,533. The largest contig showed a 7,830 base sequence not present in the consensus sequence, but matching perfectly to another (BE/43/2011) HCMV sequence, demonstrating the ability of Haploflow to accurately phase different haplotypes.

Runtime and memory consumption

Haploflow's run time depends on the three main steps (Fig. 1): first reading in the read data and building the deBruijn graph. For this every read is split into *k*-mers, with a time complexity in O(n) for the number of reads n. Since the maximal number of *k*-mers is constant in the number of reads and the length of the reads with $|k| = (\text{length}(n) - k) \cdot |n|$, it is also in O($k$). Next the graph is split into CCs and the unitig graph constructed, with a time complexity of O(k) using Tarjan's algorithm[70]. Finally the overall complexity of the assembly step is dominated by finding the paths through the unitig graph. While theoretically there is an exponential number of different paths through a graph, every vertex can only be the source of a path once and every path has length at most k, since vertices cannot be visited multiple times on the same path. The worst-case complexity of the assembly step is thus in $O(k^2)$, where k is the number of distinct *k*-mers. In practice, the number of paths is usually limited by the number of different strains, causing this step to also be linear time complexity.

For runtime assessment we compared Haploflow to SAVAGE and PEHaplo, the only other haplotype assemblers able to process the HCMV data, though SAVAGE only in *reference-based* mode, as well as metaSPAdes and MEGAHIT, which performed closest to Haploflow in terms of the summary score or is a very fast metagenome assembler, respectively (Table 1). On the HIV data, Haploflow was more than twice as fast than SAVAGE. The running time and memory requirements of Haploflow and metaSPAdes were comparable, while MEGAHIT was most efficient.

On the HIV three strain and the HCMV two strain mixtures, building the deBruijn graph and creation of the unitig graphs from the reads dominated the overall running time. For the HIV data, building the deBruijn and unitig graphs took ~8 minutes on a laptop with 4 cores and 16 GB RAM. The resulting single unitig graph included 281 vertices and assembly finished after 0.6 seconds. For the HCMV data, assembly on the same laptop required ~100 minutes, of which 85 were used for building the deBruijn and unitig graphs from the reads.

Table 1: Run time and memory consumption of Haploflow, SAVAGE in de novo mode (version 0.4.1), metaSPAdes (3.14) and MEGAHIT (1.2.9). Time and memory is averaged for the HCMV mixtures. SAVAGE did not successfully complete on the HCMV in-vitro mixtures and the simulated virome data. Values were calculated using linux' time command.

Software/ Dataset	**HIV 3 in silico mixture**		**HCMV in vitro mixture**		**Simulated virome**	
Metric	CPU *user* time (seconds)	Memory peak (GB)	Avg. CPU user time (seconds)	Avg. memory peak (GB)	CPU user time (seconds)	Memory peak (GB)
Haploflow	724	**0.009**	5,170	17.509	18,245	47.678
SAVAGE	110,208	102.938	75,518	17.658	-	-
PEHaplo	10,127	11.819	58,920	13.998	-	-
metaSPAdes	1,500	1.054	42,906	65.641	25,996	23.399
MEGAHIT	**250**	0.269	**2,910**	**0.754**	**9,690**	**2.148**

Discussion and conclusions

Viral pathogens can evolve rapidly, leading to infections with multiple strains either by within-host evolution or multiple infections of the same host. Reconstructing their genomes in a strain-resolved manner can substantially advance our understanding and capabilities to combat the diseases they cause. It is also key for genomic epidemiology, i.e. tracing viral spread using genomic information[71,72] and genome-based viral phenotyping[73]. Strains can differ in their phenotypes, such as virulence, resistance, or the degree of their immune resistance to host immunity, which may be critical for the choice of therapy.

Strain-resolved *de novo* assembly from short-read as well as long read data generated in viral genome sequencing, however, is also extremely challenging. Haploflow fills a void between fast metagenome assemblers not aiming for strain-level resolution, and viral haplotype

assemblers for small viral genomes of a few kb in size. It combines the best of both worlds for strain-resolved genome assembly, by using the fast algorithms of the metagenome assemblers, i.e. deBruijn graph based assembly, together with a specialised flow algorithm for capturing strain variation, which allows to link variants that do not co-occur on reads.

Taken together, our results demonstrate a substantial performance improvement in strain-resolved assembly for Haploflow in comparison to sixteen other metagenome and viral haplotype assemblers evaluated across different benchmark data sets. The benchmark experiments on data sets with varying numbers of strains and abundances demonstrated that Haploflow can handle data sets with substantial variation in genomic coverage introduced by amplicon sequencing and resolved strains at different degrees of evolutionary divergences well, ranging from 95% ANI (HIV), over 98% ANI (HCMV), to more than 99% ANI (SARS-CoV-2 data). On the six lab-generated HCMV mixed strain data sets, Haploflow was top scoring in the most metrics (5 of 8) in comparison to twelve other assemblers. This performance improvement in strain recall, strain precision, composite score, genome fraction and NGA50 was largely due to a better assembly of the less abundant strains. Except for Haploflow and SAVAGE **no** method assembled low abundant strains to 50% on average and Haploflow had a *far* higher NGA50, creating long contigs rather than a highly fragmented assembly. On the clinical HCMV data tested, Haploflow almost perfectly (91.7% recall and 93.6% precision) assembled strains with variants predicted by variant callers and very closely predicted the abundances of second and third strains. On a three strain HIV data set, Haploflow assembled all three genomes almost entirely, with very few mismatches. This is reflected in Haploflow scoring top in all eight metrics, with a composite assembly score of 9.66 (out of 10), in comparison to 8.02 for the best *reference-based* assembler PredictHaplo, and of 6.28 for the best *reference-free* assembler PEHaplo.

Benchmarking on a rather complex simulated virome data set with 417 taxa with unique genomes and 155 genomes in common strain taxa showed that Haploflow successfully assembled 2-3 strains for “common strain taxa” with 2-11 strains, substantially better so than the state-of-the-art metagenome assemblers able to process these data, that the tested haplotype assemblers could not. This effect was particularly pronounced for strain genome coverages within a favorable (>8) range for assembly. The abundance distribution of taxa in microbial communities is assumed to be oftentimes log-normal[13], with only a few abundant and a long tail of very low abundant ones with consequently low coverages. This indicates

that Haploflow is suitable for processing many real world data sets and characterizing the more abundant strains, similar to the reference-based StrainPhlan strain-typing software[74]. Finally, Haploflow reconstructed multiple, full length SARS-CoV-2 strains from a multi-sample wastewater metagenome data set with exact matches to clinical isolate genomes found in the GISAID database, highlighting the ability of Haploflow to recover high quality, strain-resolved viral haplotype genomes from metagenomic data.

In addition to short-read data, Haploflow also allows processing of long read data, which we demonstrated on the SARS-CoV-2 clinical data sets. For most applications dealing with low viral loads (e.g. the SARS-CoV-2 sequencing demonstrated in this article), PCR amplification is necessary to enrich viral reads. This naturally limits the possible maximum read length to the length of the PCR product, which is for those applications in the domain of short-read sequencing. The speed of the Haploflow algorithm principally also allows its extension to bacterial data, e.g. by adding multi-core and multi-*k* support and modules for handling differently sized and structured microbial genomes. Thus strain-resolved assembly from metagenome data for microbial taxa with several closely related strains could be a future application.

Availability of Data and Materials

The code of Haploflow is available on Github under https://github.com/hzi-bifo/Haploflow. The version used for the assemblies in this publication is available under the DOI https://doi.org/10.5281/zenodo.4106497. All supplemental scripts used are on Github under https://github.com/hzi-bifo/Haploflow_supplementary and are stored using publisso along all data sets and performed evaluations with the DOI TODO.

Funding

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2155 – Projektnummer 390874280.

Acknowledgements

We thank Thorsten Klingen, Susanne Reimering, Sama Goliaei and Hadi Foroughmand for helpful discussions.

Bibliography

1. Richter, M. & Rosselló-Móra, R. Shifting the genomic gold standard for the prokaryotic species definition. *Proc. Natl. Acad. Sci. U. S. A.* **106**, 19126–19131 (2009).
2. Waner, J. L. Mixed viral infections: detection and management. *Clin. Microbiol. Rev.* **7**, 143–151 (1994).
3. Ghedin, E. *et al.* Mixed Infection and the Genesis of Influenza Virus Diversity. *J. Virol.* **83**, 8832–8841 (2009).
4. Ojosnegros, S., Beerenwinkel, N. & Domingo, E. Competition-colonization dynamics: An ecology approach to quasispecies dynamics and virulence evolution in RNA viruses. *Commun. Integr. Biol.* **3**, 333–336 (2010).
5. Kumar, N., Sharma, S., Barua, S., Tripathi, B. N. & Rouse, B. T. Virological and Immunological Outcomes of Coinfections. *Clin. Microbiol. Rev.* **31**, (2018).
6. Baaijens, J. A. & Schönhuth, A. Overlap graph-based generation of haplotigs for diploids and polyploids. *Bioinformatics* **35**, 4281–4289 (2019).
7. Töpfer, A. *et al.* Viral Quasispecies Assembly via Maximal Clique Enumeration. *PLOS Comput. Biol.* **10**, e1003515 (2014).
8. Li, D., Liu, C.-M., Luo, R., Sadakane, K. & Lam, T.-W. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinforma. Oxf. Engl.* **31**, 1674–1676 (2015).
9. Nurk, S., Meleshko, D., Korobeynikov, A. & Pevzner, P. A. metaSPAdes: a new versatile metagenomic assembler. *Genome Res.* **27**, 824–834 (2017).
10. Peng, Y., Leung, H. C. M., Yiu, S. M. & Chin, F. Y. L. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinforma. Oxf. Engl.* **28**, 1420–1428 (2012).
11. Boisvert, S., Laviolette, F. & Corbeil, J. Ray: Simultaneous Assembly of Reads from a Mix of High-Throughput Sequencing Technologies. *J. Comput. Biol.* **17**, 1519–1533

(2010).

12. Simpson, J. T. *et al.* ABySS: A parallel assembler for short read sequence data. *Genome Res.* **19**, 1117–1123 (2009).
13. Sczyrba, A. *et al.* Critical Assessment of Metagenome Interpretation—a benchmark of metagenomics software. *Nat. Methods* **14**, 1063–1071 (2017).
14. Deng, Z.-L. *et al.* Evaluating assembly and variant calling software for strain-resolved analysis of large DNA-viruses. *bioRxiv* 2020.05.14.095265 (2020) doi:10.1101/2020.05.14.095265.
15. Eriksson, N. *et al.* Viral Population Estimation Using Pyrosequencing. *PLoS Comput. Biol.* **4**, (2008).
16. Astrovskaya, I. *et al.* Inferring viral quasispecies spectra from 454 pyrosequencing reads. *BMC Bioinformatics* **12**, S1 (2011).
17. Mancuso, N., Tork, B., Skums, P., Măndoiu, I. & Zelikovsky, A. Viral quasispecies reconstruction from amplicon 454 pyrosequencing reads. in *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)* 94–101 (2011). doi:10.1109/BIBMW.2011.6112360.
18. O'Neil, S. T. & Emrich, S. J. Haplotype and minimum-chimerism consensus determination using short sequence data. *BMC Genomics* **13**, S4 (2012).
19. Miller, J. R., Koren, S. & Sutton, G. Assembly Algorithms for Next-Generation Sequencing Data. *Genomics* **95**, 315–327 (2010).
20. Schatz, M. C., Delcher, A. L. & Salzberg, S. L. Assembly of large genomes using second-generation sequencing. *Genome Res.* **20**, 1165–1173 (2010).
21. Pevzner, P. A., Tang, H. & Waterman, M. S. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U. S. A.* **98**, 9748–9753 (2001).
22. Pevzner, P. A., Tang, H. & Tesler, G. De Novo Repeat Classification and Fragment Assembly. *Genome Res.* **14**, 1786–1796 (2004).

23. Bankevich, A. *et al.* SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *J. Comput. Biol.* **19**, 455–477 (2012).
24. Mohamadi, H., Chu, J., Vandervalk, B. P. & Birol, I. ntHash: recursive nucleotide hashing. *Bioinformatics* **32**, 3492–3494 (2016).
25. Compeau, P. E. C., Pevzner, P. A. & Tesler, G. Why are de Bruijn graphs useful for genome assembly? *Nat. Biotechnol.* **29**, 987–991 (2011).
26. Chor, B., Horn, D., Goldman, N., Levy, Y. & Massingham, T. Genomic DNA k-mer spectra: models and modalities. *Genome Biol.* **10**, R108 (2009).
27. Idury, R. M. & Waterman, M. S. A New Algorithm for DNA Sequence Assembly. *J. Comput. Biol.* **2**, 291–306 (1995).
28. Melsted, P. & Halldórsson, B. V. KmerStream: streaming algorithms for k -mer abundance estimation. *Bioinformatics* **30**, 3541–3547 (2014).
29. Chikhi, R. & Medvedev, P. Informed and automated k-mer size selection for genome assembly. *Bioinformatics* **30**, 31–37 (2014).
30. Dijkstra, E. W. A note on two problems in connexion with graphs. (1959).
31. luca. CS 261 Lecture 10: the fattest path. *in theory* https://lucatrevisan.wordpress.com/2011/02/04/cs-261-lecture-10-the-fattest-path/ (2011).
32. Aird, D. *et al.* Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. *Genome Biol.* **12**, R18 (2011).
33. Schirmer, M., D'Amore, R., Ijaz, U. Z., Hall, N. & Quince, C. Illumina error profiles: resolving fine-scale variation in metagenomic sequencing data. *BMC Bioinformatics* **17**, 125 (2016).
34. Sivadasan, N., Srinivasan, R. & Goyal, K. Kmerlight: fast and accurate k-mer abundance estimation. *ArXiv160905626 Cs* (2016).
35. Laehnemann, D., Borkhardt, A. & McHardy, A. C. Denoising DNA deep sequencing

data—high-throughput sequencing errors and their correction. *Brief. Bioinform.* **17**, 154–179 (2016).

36. Walker, A. *et al.* Genetic structure of SARS-CoV-2 in Western Germany reflects clonal superspreading and multiple independent introduction events. *medRxiv* 2020.04.25.20079517 (2020) doi:10.1101/2020.04.25.20079517.
37. Rose, R. *et al.* Intra-host site-specific polymorphisms of SARS-CoV-2 is consistent across multiple samples and methodologies. *medRxiv* 2020.04.24.20078691 (2020) doi:10.1101/2020.04.24.20078691.
38. Moreno, G. K. *et al.* Limited SARS-CoV-2 diversity within hosts and following passage in cell culture. *bioRxiv* 2020.04.20.051011 (2020) doi:10.1101/2020.04.20.051011.
39. Gurevich, A., Saveliev, V., Vyahhi, N. & Tesler, G. QUAST: quality assessment tool for genome assemblies. *Bioinforma. Oxf. Engl.* **29**, 1072–1075 (2013).
40. Shu, Y. & McCauley, J. GISAID: Global initiative on sharing all influenza data – from vision to reality. *Eurosurveillance* **22**, (2017).
41. Wilm, A. *et al.* LoFreq: a sequence-quality aware, ultra-sensitive variant caller for uncovering cell-population heterogeneity from high-throughput sequencing datasets. *Nucleic Acids Res.* **40**, 11189–11201 (2012).
42. Holmes, E. C. *The Evolution and Emergence of RNA Viruses*. (Oxford University Press, 2009).
43. Crits-Christoph, A. *et al.* Genome sequencing of sewage detects regionally prevalent SARS-CoV-2 variants. *medRxiv* 2020.09.13.20193805 (2020) doi:10.1101/2020.09.13.20193805.
44. Olm, M. R. *et al. InStrain enables population genomic analysis from metagenomic data and rigorous detection of identical microbial strains*. http://biorxiv.org/lookup/doi/10.1101/2020.01.22.915579 (2020) doi:10.1101/2020.01.22.915579.

45. Li, H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* **32**, 2103–2110 (2016).
46. Caporaso, J. G. *et al.* Ultra-high-throughput microbial community analysis on the Illumina HiSeq and MiSeq platforms. *ISME J.* **6**, 1621–1624 (2012).
47. Hesse, U. *et al.* Virome Assembly and Annotation: A Surprise in the Namib Desert. *Front. Microbiol.* **8**, 13 (2017).
48. Sutton, T. D. S., Clooney, A. G., Ryan, F. J., Ross, R. P. & Hill, C. Choice of assembly software has a critical impact on virome characterisation. *Microbiome* **7**, 12 (2019).
49. Hage, E. *et al.* Characterization of Human Cytomegalovirus Genome Diversity in Immunocompromised Hosts by Whole-Genome Sequencing Directly From Clinical Specimens. *J. Infect. Dis.* **215**, 1673–1683 (2017).
50. Mikheenko, A., Saveliev, V. & Gurevich, A. MetaQUAST: evaluation of metagenome assemblies. *Bioinformatics* **32**, 1088–1090 (2016).
51. van der Kuyl, A. C. & Cornelissen, M. Identifying HIV-1 dual infections. *Retrovirology* **4**, 67 (2007).
52. Leye, N. *et al.* High frequency of HIV-1 infections with multiple HIV-1 strains in men having sex with men (MSM) in Senegal. *Infect. Genet. Evol. J. Mol. Epidemiol. Evol. Genet. Infect. Dis.* **20**, 206–214 (2013).
53. Baaijens, J. A., Aabidine, A. Z. E., Rivals, E. & Schönhuth, A. De novo assembly of viral quasispecies using overlap graphs. *Genome Res.* **27**, 835–848 (2017).
54. Chen, J., Zhao, Y. & Sun, Y. De novo haplotype reconstruction in viral quasispecies using paired-end read guided path finding. *Bioinforma. Oxf. Engl.* **34**, 2927–2935 (2018).
55. Brister, J. R., Ako-Adjei, D., Bao, Y. & Blinkova, O. NCBI viral genomes resource. *Nucleic Acids Res.* **43**, D571-577 (2015).
56. Fritz, A. *et al.* CAMISIM: simulating metagenomes and microbial communities. *Microbiome* **7**, 17 (2019).

57. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinforma. Oxf. Engl.* **25**, 2078–2079 (2009).

58. Ke, Z. & Vikalo, H. A Graph Auto-Encoder for Haplotype Assembly and Viral Quasispecies Reconstruction. *Proc. AAAI Conf. Artif. Intell.* **34**, 719–726 (2020).

59. Hage, E. *et al.* Characterization of Human Cytomegalovirus Genome Diversity in Immunocompromised Hosts by Whole-Genome Sequencing Directly From Clinical Specimens. *J. Infect. Dis.* **215**, 1673–1683 (2017).

60. Sijmons, S., Van Ranst, M. & Maes, P. Genomic and Functional Characteristics of Human Cytomegalovirus Revealed by Next-Generation Sequencing. *Viruses* **6**, 1049–1072 (2014).

61. Hunt, M. *et al.* IVA: accurate de novo assembly of RNA virus genomes. *Bioinforma. Oxf. Engl.* **31**, 2374–2376 (2015).

62. Yang, X. *et al.* De novo assembly of highly diverse viral populations. *BMC Genomics* **13**, 475 (2012).

63. Fedonin, G. G., Fantin, Y. S., Favorov, A. V., Shipulin, G. A. & Neverov, A. D. VirGenA: a reference-based assembler for variable viral genomes. *Brief. Bioinform.* **20**, 15–25 (2017).

64. Prabhakaran, S., Rey, M., Zagordi, O., Beerenwinkel, N. & Roth, V. HIV haplotype inference using a propagating dirichlet process mixture model. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **11**, 182–191 (2014).

65. Knyazev, S. *et al.* CliqueSNV: An Efficient Noise Reduction Technique for Accurate Assembly of Viral Variants from NGS Data. *bioRxiv* 264242 (2020) doi:10.1101/264242.

66. Splettstoesser, T. *English: Structure of the HIV-1 genome. It has a size of roughly 10.000 base pairs and consists of nine genes, some of which are overlapping.* (2014).

67. Mikheenko, A., Valin, G., Prjibelski, A., Saveliev, V. & Gurevich, A. Icarus: visualizer for de novo assembly evaluation. *Bioinformatics* **32**, 3321–3323 (2016).

68. Olm, M. R., Brown, C. T., Brooks, B. & Banfield, J. F. dRep: a tool for fast and accurate genomic comparisons that enables improved genome recovery from metagenomes through de-replication. *ISME J.* **11**, 2864–2868 (2017).

69. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

70. Tarjan, R. E. Depth-First Search and Linear Graph Algorithms. *SIAM J Comput* (1972) doi:10.1137/0201010.

71. Grenfell, B. T. *et al.* Unifying the epidemiological and evolutionary dynamics of pathogens. *Science* **303**, 327–332 (2004).

72. Reimering, S., Muñoz, S. & McHardy, A. C. Phylogeographic reconstruction using air transportation data and its application to the 2009 H1N1 influenza A pandemic. *PLOS Comput. Biol.* **16**, e1007101 (2020).

73. Beerenwinkel, N. *et al.* Geno2pheno: Estimating phenotypic drug resistance from HIV-1 genotypes. *Nucleic Acids Res.* **31**, 3850–3855 (2003).

74. Truong, D. T., Tett, A., Pasolli, E., Huttenhower, C. & Segata, N. Microbial strain-level population structure and genetic diversity from metagenomes. *Genome Res.* **27**, 626–638 (2017).

75. Tørresen, O. K. *et al.* Tandem repeats lead to sequence assembly errors and impose multi-level challenges for genome and protein databases. *Nucleic Acids Res.* **47**, 10994–11006 (2019).

76. Guo, Y. *et al.* The effect of strand bias in Illumina short-read sequencing data. *BMC Genomics* **13**, 666 (2012).

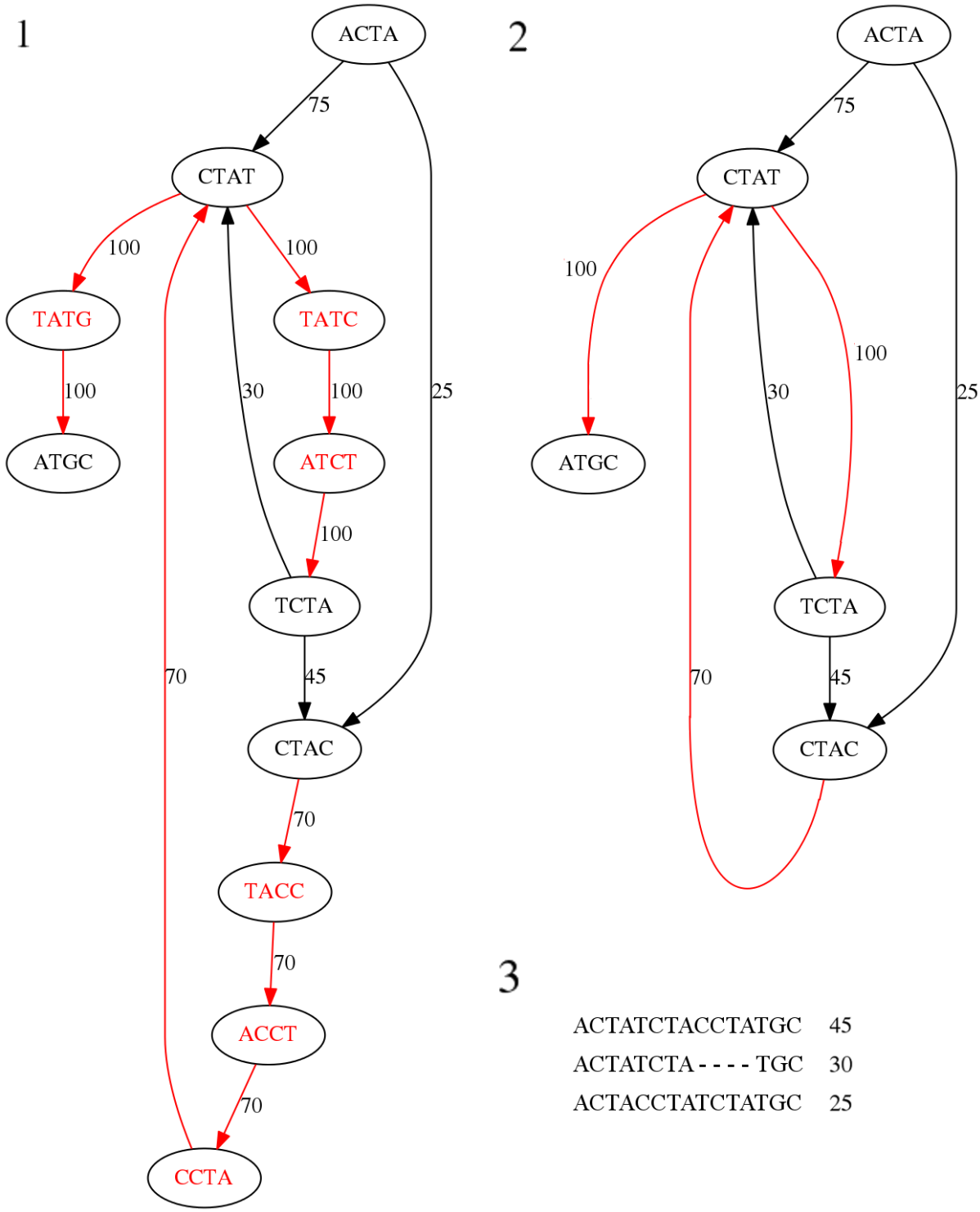
77. A reference standard for genome biology. *Nat. Biotechnol.* **36**, 1121–1121 (2018).

78. Suárez, N. M. *et al.* Human Cytomegalovirus Genomes Sequenced Directly From Clinical Material: Variation, Multiple-Strain Infection, Recombination, and Gene Loss. *J. Infect. Dis.* **220**, 781–791 (2019).

Supplement

Exemplary clarification of path finding step realized in Haploflow

In the unitig graph there are multiple paths between a source and a sink which (sans sequencing errors) correspond to the different strains present in a sample. The choice of the correct path follows the fatness algorithm described before. There is another factor though, namely the length of the fattest path, which Haploflow *also* maximises. In Figure S1 there is exactly one source, the vertex ACTA, and one sink, the vertex ATGC, but there are infinitely many paths from ACTA to ATGC, since CTAT to TCTA and TCTA to CTAT form a loop. To prevent this, Haploflow allows every edge only to be used *once* in every path finding step. This makes the particular loop in Figure S1 "resolvable", the number of paths reduces to five:
1: $ACTA \rightarrow CTAT \rightarrow TCTA \rightarrow CTAT \rightarrow ATGC$ with a fatness of 30
2: $ACTA \rightarrow CTAT \rightarrow TCTA \rightarrow CTAC \rightarrow CTAT \rightarrow ATGC$ with a fatness of 45
3: $ACTA \rightarrow CTAT \rightarrow ATGC$ with a fatness of 75
4: $ACTA \rightarrow CTAC \rightarrow CTAT \rightarrow ATGC$ with a fatness of 25
5: $ACTA \rightarrow CTAC \rightarrow CTAT \rightarrow TCTA \rightarrow CTAT \rightarrow ATGC$ with a fatness of 25
Just going by the fattest graph, path 3 would get selected, but this path is shorter than all other paths and thus only paths 2 and 5 can be selected, out of which path 2 has the higher fatness of 45 (the coverage of the first sequence). The next longest and fattest path is path 5 with a fatness of 25 (the coverage of the last sequence) and finally path 1 remains with a fatness of 30. Paths 3 and 4 do not exist at this point, since the capacity of all edges has been used.



Suppl. Figure S1: The deBruijn graph (1) and its corresponding Unitig graph (2) for three related sequences and their coverage (3). The red *k*-mers and edges between them are part of linear paths and are replaced by a single red edge in the unitig graph. The edges are labelled with the "capacity", the sum of the coverages of the sequences going over them, in the deBruijn graph and the average capacity of all smoothed edges in the Unitig graph - which in

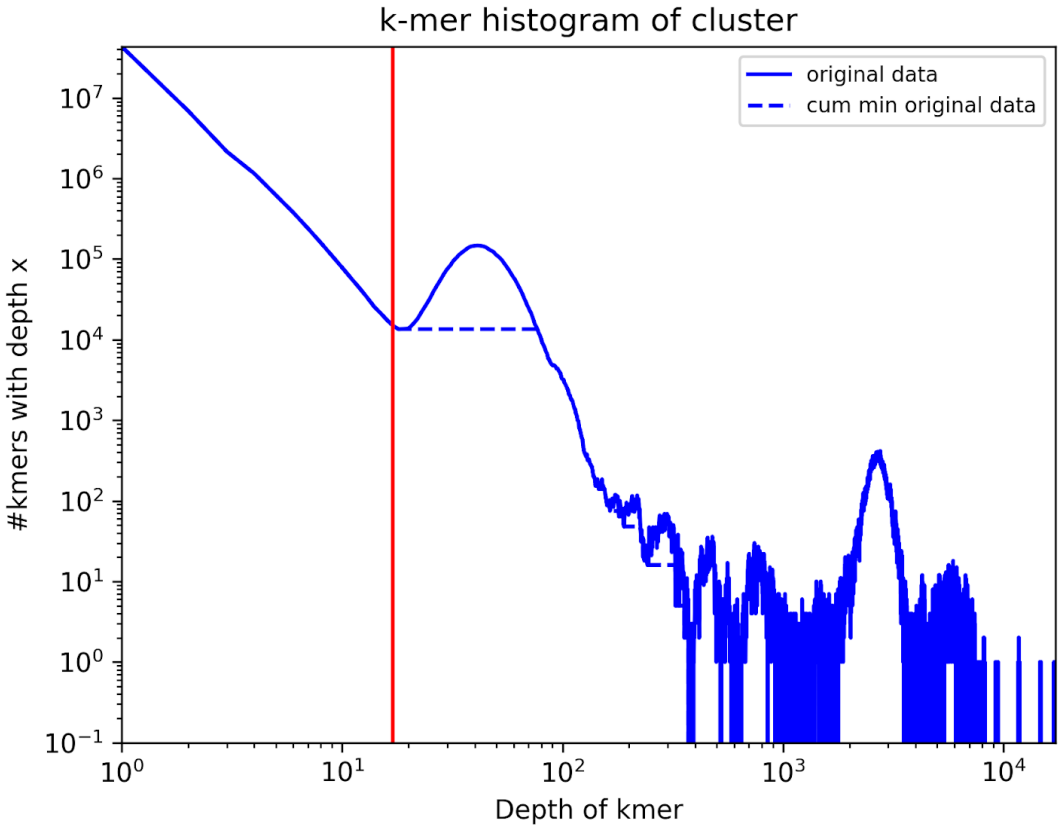
this case is the same as the original capacity. Some of the edges represent one (capacities 25, 30, 45), some two (capacity 70 = 45 + 25) and some all (capacity 100 = 45 + 30 + 25) of the sequences.

Algorithmic details of the flow algorithm

The fatness of a path is defined by the lowest fatness value of any edge along this path. Since the fatness of an edge might be underestimated if the coverage dropped for edges occurring before this edge in the path, it is not sufficient to just remove the calculated fatness when reducing flow along a path. Instead, the coverage of the source is set to 0 and for every other edge on the path the flow is reduced to *max(capacity - previously_removed_flow, 0)* where *previously_removed_flow* is the flow removed from the last edge on the path. Since it is possible that edges are used multiple times, it is also possible that there are paths that have hardly any edges that are "unique" to that path. We call an edge *unique*, if it is part of exactly one path. If the fraction or length of unique edges of a path is too low, by default less than 500 bases, the path is removed for all edges on which it is not unique, to avoid overestimating the total number of paths in the graph. Edges with coverage of 0 will get removed, possibly producing new sources. If Haploflow crosses a junction with two or more outgoing edges with similar coverage values and cannot make an informed decision which is the higher abundant path, Haploflow will break the contig at this position. This happens either if multiple strains have very similar coverages or on genomic repeats. The exact threshold for this break is derived using the *error_rate* and *strict/threshold* parameter: If the difference is less than the percentage value given or the (either explicitly stated or derived from *strict*) threshold, the contig is broken.

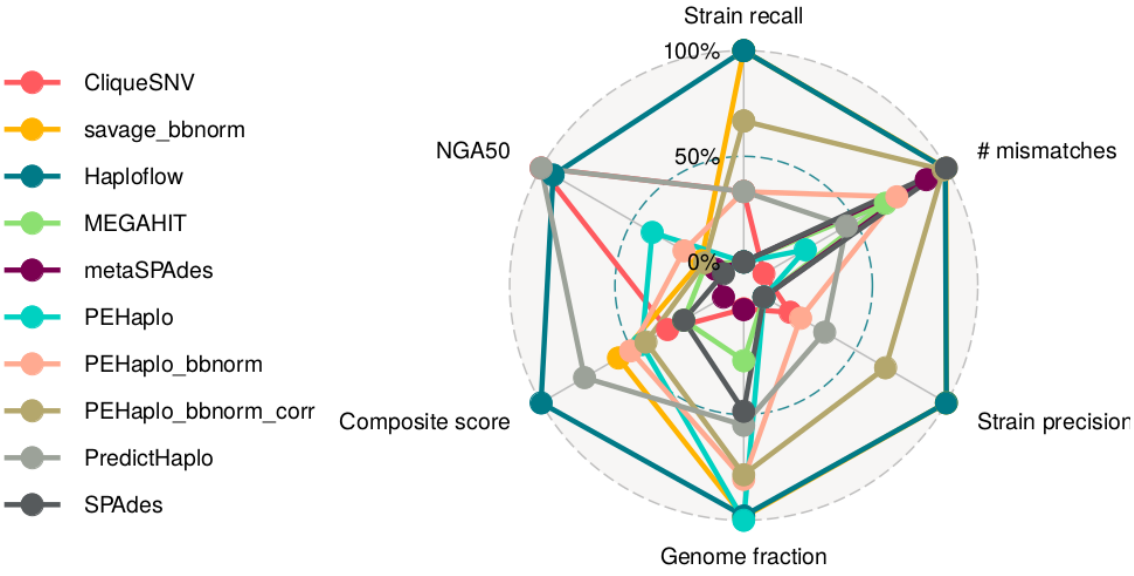
After the path has been found, the coverage of all unique edges on this path is reduced to 0, as no other path will be traversing this edge. If there is more than one path going over the edge, then the flow is reduced, corresponding to the expected coverage of the current edge. This value is the flow removed from the last visited unique edge, meaning that local increases and decreases in coverage are also captured. If the coverage of an edge would be reduced to 0, even though there are still paths going over this edge, the coverage is set to a dummy value such that it can still be used. On the other hand, if a path consists solely of non-unique edges, a duplication is assumed and the current path is not considered.

When permanently reducing the flow, it is not sufficient to remove the (overall) fatness of the path, since the fatness can only decrease (or stay the same) along a path, while the coverage values might fluctuate, based on amplification and sequencing strategy. To circumvent this, the flow is reduced by a "local fatness": All unique edges are removed as described before, for all other edges either flow removed from the last edge or, if the value is higher, of the average per-base removed flow, is taken as a baseline and depending on the fact whether the flow decreased or increased within the last edge, the flow to be removed is decreased and increased accordingly. If there would not be any flow remaining, a minimal value is left over.

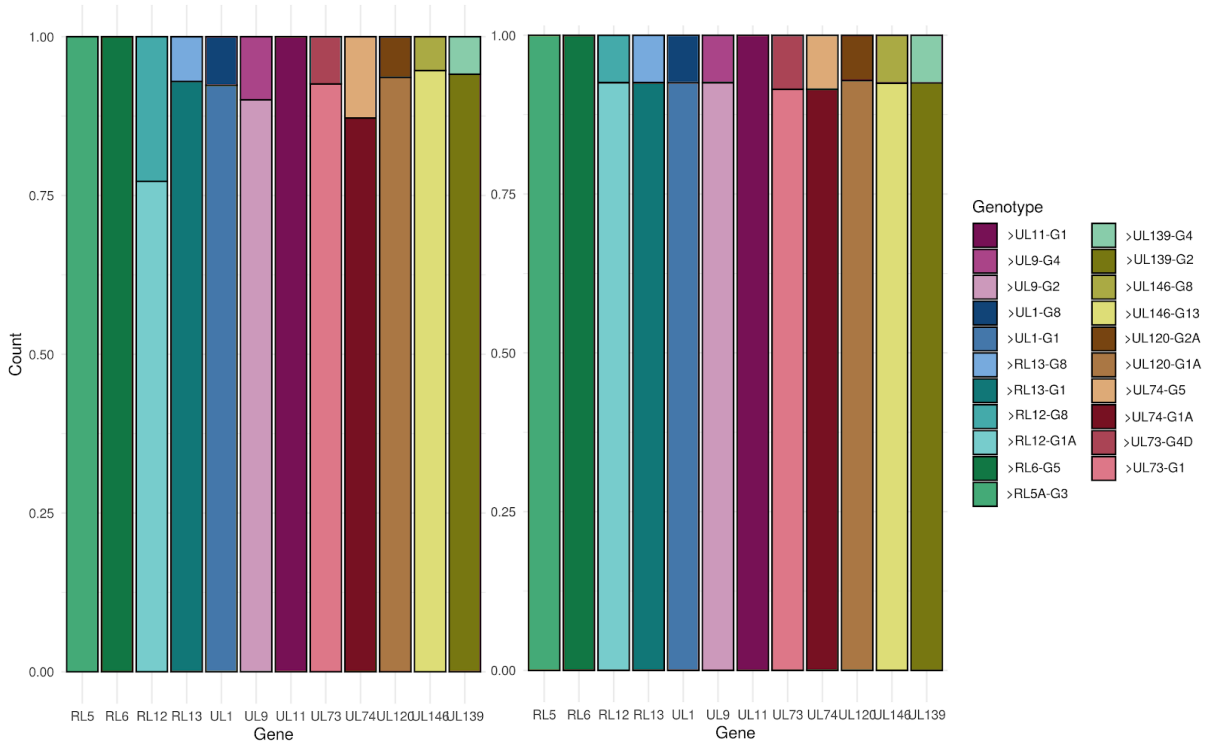


Suppl. Figure S2: log-log kmer coverage histogram for a sequence sample of HCMV strain TB40E and *E. coli.* This shows the kmer coverage (or counts) on the x-axis versus the number of kmers with that coverage on the y-axis. Original values are shown with the solid line, the cumulative minimum of the number of k-mers of a certain depth with the dashed line. k-mers with a depth less than the depth of the first k-mer for which the cumulative minimum (cum min) is less than the original value are regarded as probable erroneous k-mers (red line). For example, for a mixture of *Escherichia coli* (5,129,110 bp) and HCMV

(234,127 bp), with a length ratio of 22:1, distinct peaks occur at coverages of ~45 and ~2500. The first peak has 10,000 distinct kmers and the second one 400, indicating that the first genome might be around 25x as large as the second one.



Suppl. Figure S3: Radar plot of relative performance for Haploflow and nine other methods for the HIV-3 *in silico* data set. Best performance is at 100% and Haploflow, in dark blue, ranks first in Strain recall, Strain precision and composite score.



Suppl. Figure S4: Genes with different genotypes and their coverage distribution in the reads (left) and in the contigs (right). VATK was run on the reads and the contigs. For the contigs the bar height was set based on the coverage value Haploflow reports. VATK found the exact same genotypes in the reads and the Haploflow contigs.

Reconstruction of full length SARS-CoV-2 sequences

In nine out of 17 SARS-CoV-2 samples and 6 out of 7 wastewater SARS-CoV-2 samples, quast reported a high duplication ratio for the Haploflow assembly; four out of five DUS and five out of twelve WIS samples. This can be explained by either artificially duplicating parts of the genome or the presence of two closely related strains. Since Haploflow did not construct single contig assemblies for all these strains, first a "scaffolding" step was performed: All contigs are clustered using *k*-means clustering on Haploflow's predicted abundance, the number of clusters depending on the duplication ratio. Then, using the NC_0455122.2 RefSeq strain, the contigs are extended to complete genomes, using the contigs bases for all parts of the genome covered by it. If a part of a genome is not covered by a strain, the bases from the highest abundant strain with bases at this position are inserted, if

no strain has bases at this position, the reference base is inserted. If a position is covered twice, the base from the contig with higher flow is chosen. To reduce the number of false positive SNPs, an additional filtering step was performed to remove typical sequencing and PCR related artifacts, such as deletions within homopolymeric sites[35], mutations in short-tandem repeats[75] and mutations on sites with strong strand bias[76]. This for example removed a SNP included in the original submission of the DUS sample (in the HCMV data set) at position 4655 due to a high strand bias value.

Lofreq version 2.1.4 was run on the original reads and the variants filtered by an abundance value over 5% and a score of >1000 to reduce the number of false positive calls. This filtered similar SNPs as the filtering of homopolymeric or strand biased sites performed for Haploflow.

Supplementary tables

Suppl. Table S1: Benchmark of Haploflow against five *de novo* assemblers and five reference-based assemblers (grey background) on the HIV-3 data set. For every metric, best performing methods (95-100% range of results) are indicated. *Strain recall*: fraction of correctly recovered high quality strain genomes (≤ 1 (≤ 5) mismatches per kb; more than 90% (80%) genome fraction[77]); *Strain precision*: fraction of correctly recovered high quality strain genomes of all genome assemblies. Evaluation using metaQUAST results and derived strain assembly metrics with HIV reference genomes 89.6, HXB-2 and JR-SCF in "combined reference" mode. [a]Results for a 140 Mb subset of the 500 Mb dataset generated with *BBnorm*. *runs that did not complete after ten days or failed. **as being an outlier, QuasiRecomb results were excluded from composite score and radar plot calculation.

	Strain recall	**Strain precision**	**Compo-site score**	**Genome fraction (%)**	**Number of contigs**	**Mis-matches**	**Dupli-cation ratio**	**NGA50**
Haploflow	**3/3**	**3/3**	**9.66**	**93.36**	**3**	**9**	**1.001**	**9083**
metaSPAdes	0/3	0/3	2.78	33.70	9	62	**1.005**	1246
SPAdes	0/3	0/3	4.28	63.09	20	**1**	**1.021**	864
MEGAHIT	0/3	0/3	4.28	48.47	12	257	**1.084**	1701
PEHaplo *l=40*	0/3	0/9	5.86	**94.63**	40	1208	3.106	4305
PEHaplo[a]	0(1)/3	1/5	6.28	82.80	20	363	1.683	2774
PEHaplo[a] *correct*	0(2)/3	2/3	5.7	81.45	17	**24**	**1.044**	1773
SAVAGE *de novo**	-	-	-	-	-	-	-	-
SAVAGE *ref**	-	-	-	-	-	-	-	-
SAVAGE[a] *ref*	*2(3)/3*	*3/3*	6.74	*93.72*	18	*5*	*1.057*	1932
PredictHaplo	1/3	1/3	8.02	67.12	3	631	1.497	*9658*
QuasiRecomb **	0/3	0/3,507	-	67.12	3506	1,011,054	1748.6	*9649*
QuasiRecomb *conservative* **	0/3	0/1,155	-	67.12	1154	411,576	575.67	*9654*

CliqueSNV	1/3	1/7	4.89	33.5	7	538	7.000	*9669*
GAEseq*	-	-	-	-	-	-	-	-
GAEseq[a]*	-	-	-	-	-	-	-	-

Suppl. Table S2: Benchmark results for the HCMV data set. Shown are average values for the metaQUAST metrics over the six data sets and additional assembly metrics (see "performance evaluation"). For every metric, the best performing methods (95-100% range of results) are indicated. **Strain recall* includes correctly recovered genomes at two quality levels: more than 80(90)% genome fraction and less than 5(1) mismatches/kb. PEHaplo did not assemble one (TA-1-10) of the six mixtures.

	Strain recall*	**Strain precision**	**Compo-site score**	**Genome fraction**	**Contigs**	**Mis-matches per 100kb**	**Dupli-cation ratio**	**NGA50**
Haploflow	**10(3)/12**	**10/14**	**9.34**	**83.87 ± 10.37%**	20.50 ± 7.20	166.26 ± 122.97	1.20 ± 0.15	**62,560.42 ± 35,233**
metaSPAdes	5(4)/12	5/12	8.18	58.57 ± 4.44%	17.42 ± 8.38	184.13 ± 337.14	**1.01 ± 0.01**	**60,008.25 ± 37,089**
SPAdes	6(4)/12	6/12	5.22	65.52 ± 3.94%	85.17 ± 15.75	40.38 ± 29.84	1.05 ± 0.01	2,552.42 ± 951
MEGAHIT	2(0)/12	2/23	4.71	68.01 ± 8.23%	324.83 ± 244.29	2254.09 ± 1901.7	1.92 ± 0.67	32,446.08 ± 35,925
PEHaplo	4(3)/12	4/12	5.70	52.72 ± 18.07%	54.0 ± 78.17	**13.04 ± 11.68**	1.05 ± 0.07	10,960.1 ± 6,602
tadpole	1(0)/12	1/12	3.15	24.47 ± 13.31%	39.92 ± 12.98	27.14 ± 50.67	**1.00 ± 0.00**	1,344.3 ± 3,292
ABySS	6(3)/12	6/12	6.41	64.88 ± 4.94%	20.92 ± 8.85	250.0 ± 85.39	1.05 ± 0.01	12,399.25 ± 5,157
Ray	4(4)/12	4/12	5.90	51.39 ± 2.27%	16.67 ± 12.53	67.54 ± 63.39	1.07 ± 0.06	26,154.75 ± 36,557
IDBA	0(0)/12	0/12	3.10	32.71 ± 9.52%	83.75 ± 13.24	104.46 ± 76.39	**1.03 ± 0.01**	154.67 ± 178
Vicuna	4(1)/12	4/12	4.18	47.26 ± 0.93%	36.33 ± 7.79	104.05 ± 71.31	**1.02 ± 0.01**	2,657.67 ± 704

IVA	4(3)/12	4/12	7.42	43.23 ± 16.2%	**11.92** ± **8.22**	121.61 ± 185.89	**1.02** ± **0.03**	**63,773.0** ± **49,449**
VirGenA	*6(5)/12*	6/12	7.63	47.25 ± 1.35%	*5.67* ± *2.48*	102.41 ± 107.01	*1.01* ± *0.00*	33,324.58 ± 30,049
SAVAGE	*9(5)/12*	9/17	4.61	*82.43* ± *15.69%*	283.17 ± 98.89	33.86 ± 28.37	1.46 ± 0.14	1,245.33 ± 349

Suppl. Table S3: Genome fraction and NGA50 and their standard deviation for the high and low abundant strains in the HCMV in-vitro mixtures (two 1:10 and two 1:50 mixtures). PEHaplo and SAVAGE as reference-free haplotype assemblers did not return any results on this data set.

	metaSPAdes	MEGAHIT	Haploflow
Genome fraction (lower abundance)	19.77 ± 6.44%	30.00 ± 12.6%	**76.67 ± 12.24%**
Genome fraction (higher abundance)	**94.86 ± 0.57%**	89.97 ± 8.59%	91.62 ± 5.38%
NGA50 (lower abundance)	0 ± 0	0 ± 0	**9,625 ± 6,578**
NGA50 (higher abundance)	**149,712 ± 48,744**	45,481 ± 45,766	79,284 ± 50,679

Suppl. Table S4: One cluster with closely 11 related phage strains from the simulated virome[48] and the genome within-cluster similarities. Columns give the maximal or average ANI to all other sequences in the cluster.

GI number	Scientific name	Within-cluster similarity max	Within-cluster similarity avg
118725053	Staphylococcus phage phiNM3	98.10%	93.32%
119443652	Staphylococcus phage phiPVL108	98.31%	94.59%
157102936	Staphylococcus prophage tp310-1	99.12%	95.47%
157102938	Staphylococcus prophage tp310-3	98.74%	95.72%
239507361	Staphylococcus phage phiPVL-CN125	98.31%	94.59%
257136356	Staphylococcus phage P954	96.12%	92.18%
29028667	Staphylococcus prophage phi 13	98.74%	95.73%
30043925	Staphylococcus prophage phiN315	98.10%	92.98%
41189515	Staphylococcus phage 77	96.12%	93.13%
9635165	Staphylococcus phage PVL	99.12%	95.25%
9635677	Staphylococcus prophage phiPV83	96.59%	93.86%

Suppl. Table S5: Genome fractions in % on different subsets of the simulated virome. Unique genomes refers to genomes for which no other genome with an ANI of >95% is in the data set, common strain genomes are ones for which at least one such genome is present. The coverage value was calculated by dividing the total number of base pairs in the reads belonging to the genome by its size.

	SPAdes	MEGAHIT	Haploflow
Common strains	55.58	48.88	**62.85**
Common strains, coverage > 8	74.58	64.99	**89.36**
Total	**72.2**	68.6	66.6
Total, coverage > 8	93.07	87.55	**94.52**

Suppl. Table S6: Comparison of multiple strain infection labeling of samples by VATK[78], the predicted relative abundance of the low abundant strain(s) and the predicted abundance by Haploflow (relative and absolute) as well as the genome completeness (genome fraction, mapped against the first sample consensus genome) of strains Haploflow reconstructed (Supplementary methods). A ”-” denotes that no evidence of a second strain was found by either VATK (column 3) or Haploflow (column 4). Percentage values with a (*) denote problems in clustering, evident by a still high duplication ratio after clustering or the sum of genome fractions of two clusters summing up to ~1, indicating that underclustering or in the latter case overclustering took place. Three percentage values in the third column indicate that Haploflow predicted three strains being present.

Patient	Time points	Estimated low strain abundance (number of predicted strains)	Haploflow low strain abundance predictions (% and absolute value(s))	Genome fraction (portion of recovered genome) of strains vs. consensus sequence
RTR3	367 days	38.3% (2)	27.1% (43:16)	88.29% / 40.70%
	408 days	17.9% (2)	17.3% (167:35)	92.93% / 68.76%
RTR6	vitreous humour	9.4% (2)	12.7% (2576:373)	90.07% / 31.06%
	blood	17.3% (2)	18.4% (213:48)	99.07% / 92.73%
SCTR1	91 days	4.0% (2)	- (425)	99.69%
	126 days	22.9% (2)	21.2% (2429:653)	96.42% / 75.93%
	130 days	14.0% (2)	12.6% (313:45)	99.63% / 82.87%
	194 days	6.8% (2)	- (105)	99.78%
	224 days	37.8% (2)	26.8% (60:22)	92.73% / 79.36%
	231 days	25.4% (2)	23.6% (126:39)	92.09% / 37.63%
	244 days	- (1)	17.3% (81:17)	93.19%
	245 days	29.8% (2)	- (22)	92.86%
SCTR3	189 days	- % (1)	- (9)	98.74%
	272 days	20.5% (2)	24.3% (2781:893)	80.89% / 34.96%
	320 days	28.1% (2)	- (38)	98.46%

SCTR8	55 days	24.4% (2)	7.3% (178:14)	99.28% / 24.28%
	287 days	16.5% (2)	30.2%/8.3% (305:150:41)	93.86% / 58.5% / 52.33% (*)
SCTR11	88 days	15.6% (2)	16.3% (103:20)	99.07% / 85.76%
	192 days	11.6% (2)	11.2% (119:15)	91.75% / 78.20 %
SCTR17	21 days	34.7% (2)	22.1% (106:30)	94.81% / 30.96%
	28 days	30.6% (2)	17.6% (404:86)	98.39% / 17.04% (*)
	35 days	33.0% (2)	29.8%/9.5% (362:178:57)	99.46% / 28.08% / 22.61% (*)
	50 days	6.1% (2)	8.5% (4063:376)	97.26% / 94.18%
SCTR18	28 days	30.6% (3)	26.8% (101:37)	99.27% / 50.60% (*)
	35 days	30.3%/10.3% (3)	32.6%/10.2% (291:166:52)	80.79% / 81.88% / 48.85%
Summary strain detection		Total strains: 48	Recall: 91.7% (44/48) Precision: 93.6% (44/47)	

Suppl. Table S7: SNPs and short indels detected by Haploflow and Lofreq for all 17 samples of SARS-CoV-2 in at least one sample. Lofreq was run with default parameters and SNPs were filtered by a score of >1000 and abundance >5%. "Rare" indicates thatLofreq predicted this variant at less than 5%, homopolymeric means that this site is located within or in the direct vicinity of a 4bp or longer homopolymer.

Position	Original base	Detected base	Detected by	Notes
518-520	ATG	---	Haploflow	
686-694	AAGTCATTT	---------	Haploflow	
1440	G	A	Haploflow, Lofreq	
2891	G	A	Haploflow	
4802	G	A	Lofreq	homopolymeric
7717	T	A	Haploflow, Lofreq	
10507	C	T	Haploflow, Lofreq	
11335	G	T	Haploflow, Lofreq	
11454	C	T	Haploflow, Lofreq	
11467	G	T	Haploflow, Lofreq	
11514	C	T	Haploflow, Lofreq	
11897	C	A	Haploflow, Lofreq	
12071	G	A	Haploflow	rare
13115	C	T	Haploflow, Lofreq	
15139	A	C	Haploflow, Lofreq	
15157	C	A	Lofreq	Strand bias, called by Haploflow then filtered
15168	G	A	Haploflow, Lofreq	homopolymeric
16954	C	T	Haploflow, Lofreq	
18110	C	A	Haploflow	rare
17373	C	T	Haploflow, Lofreq	

19182	A	G	Haploflow, Lofreq	
19610	C	T	Haploflow, Lofreq	
20298-20300	ATT	---	Haploflow	
21077	C	T	Lofreq	homopolymeric
21575	C	T	Haploflow	homopolymeric
22323	C	T	Haploflow, Lofreq	
25658	C	T	Haploflow, Lofreq	
29659	C	T	Lofreq	homopolymeric
29760	T	C	Lofreq	

Suppl. Table S8: Number of SARS-Cov-2 genomes assembled by Haploflow from seven SARS-CoV-2 wastewater metagenome samples and GISAID IDs of identical genomes recovered from clinical isolates. Strains are listed in order of their estimated abundances for individual samples.

Sample	# strains Haploflow	GISAID matches to strains
Oakland 5/19	2	hCoV-19/USA/LA-SR0328/2020 hCoV-19/USA/CA-CSMC67/2020
Oakland 5/19 (2)	2	hCoV-19/Poland/PL_P31/2020 hCoV-19/Beijing/DT-BJ01/2020
Oakland 5/28	2	hCoV-19/USA/CA-CSMC25/2020 hCoV-19/USA/CA-CSMC67/2020
Oakland 6/09	1	hCoV-19/France/IDF-10064DR/2020
Oakland 6/30	2	hCoV-19/USA/WA-UW-11903/2020 hCoV-19/France/IDF-10064DR/2020
Oakland 6/30 (2)	2	hCoV-19/USA/CA-CSMC25/2020 hCoV-19/USA/LA-SR0328/2020
Marin 7/1	2	hCoV-19/USA/VA-DCLS-1271/2020 hCoV-19/USA/WA-UW-11903/2020

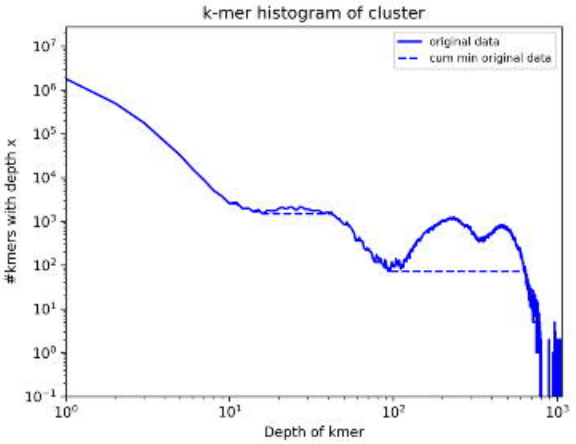
Reads

deBruijn graph

split by CCs

calculate coverages

Unitig graphs



Coverage histograms

calculate thresholds

calculate paths

calculate flow

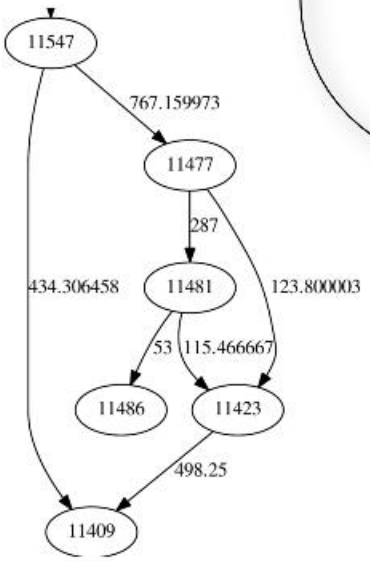
clean graph

Assembly graphs

vitreous.fa

```
>Contig_0 (0.63852 of 1704.1)
CTTTTCTCCCCATGACAGAGGAGGAGACCCCGCACCGTCCGTC
>Contig_1 (0.603666 of 615.998)
CTTTTCTCCCCATGACAGAGGAGGAGACCCCGCACCGTCCGTC
>Contig_2 (0.50835 of 244.141)
CTTTTCTCCCCATGACAGAGGAGGAGACCCCGCACCGTCCGTC
>Contig_3 (0.525521 of 121.985)
CTTTTCTCCCCATGACAGAGGAGGAGACCCCGCACCGTCCGTC
>Contig_4 (0.527525 of 57.8793)
CTTTTCTCCCCATGACAGAGGAGGAGACCCCGCACCGTCCGTC
>Contig_5 (1 of 27.3465)
CTTTTCTCCCCATGACAGAGGAGGAGACCCCGCACCGTCCGTC
>Contig_6 (0.783277 of 703.197)
CAACCGGTTAAAAATCTCACTATGAACATGACAGAGTTTCCAC
>Contig_7 (0.539485 of 152.399)
CAACCGGTTAAAAATCTCACTATGAACATGACAGAGTTTCCAC
>Contig_8 (0.517127 of 70.1819)
CAACCGGTTAAAAATCTCACTATGAACATGACAGAGTTTCCAC
>Contig_9 (0.597722 of 33.8889)
CAACCGGTTAAAAATCTCACTATGAACATGACAGAGTTTCCAC
>Contig_10 (1 of 13.6328)
CAACCGGTTAAAAATCTCACTATGAACATGACAGAGTTTCCAC
```

Contigs

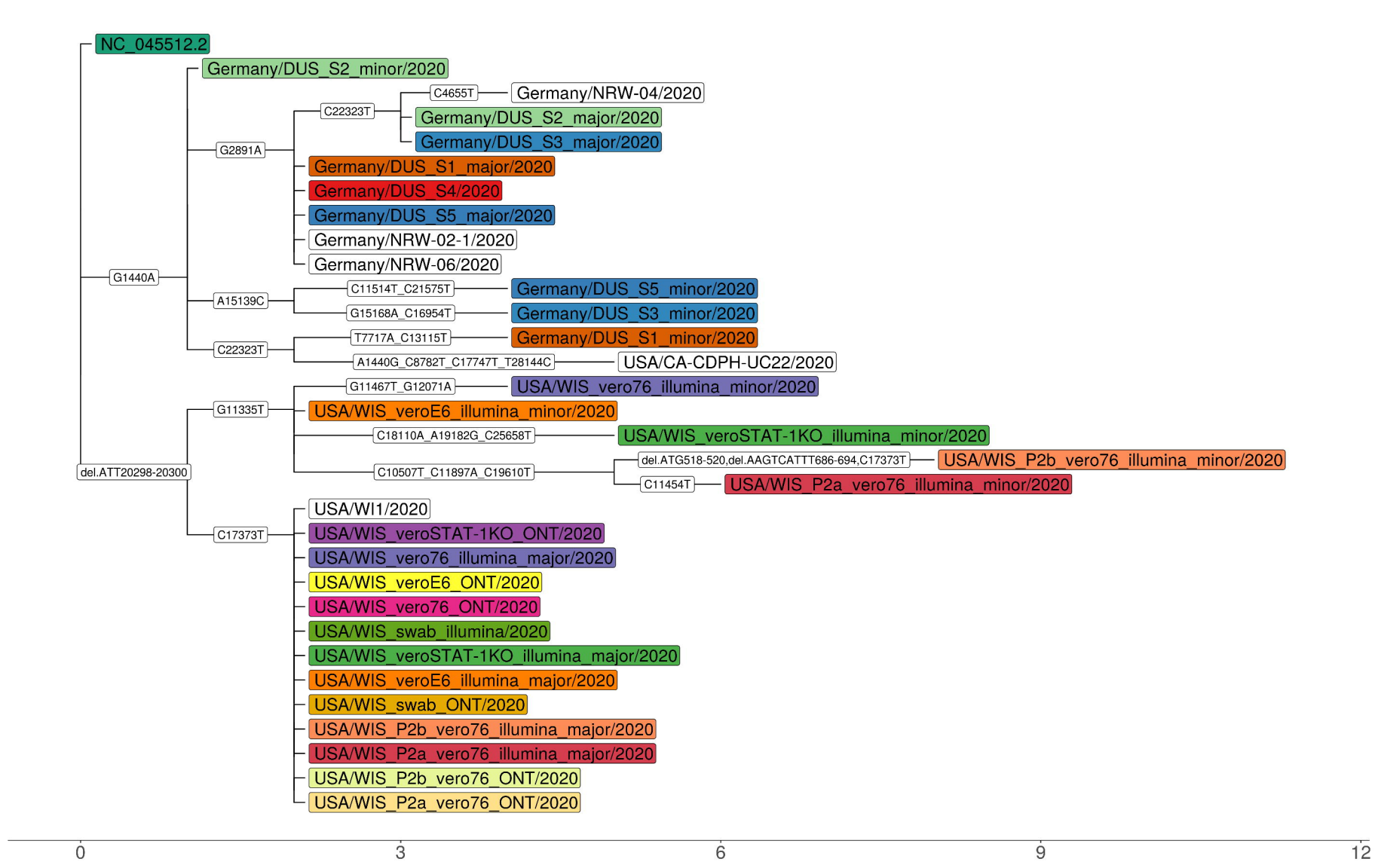


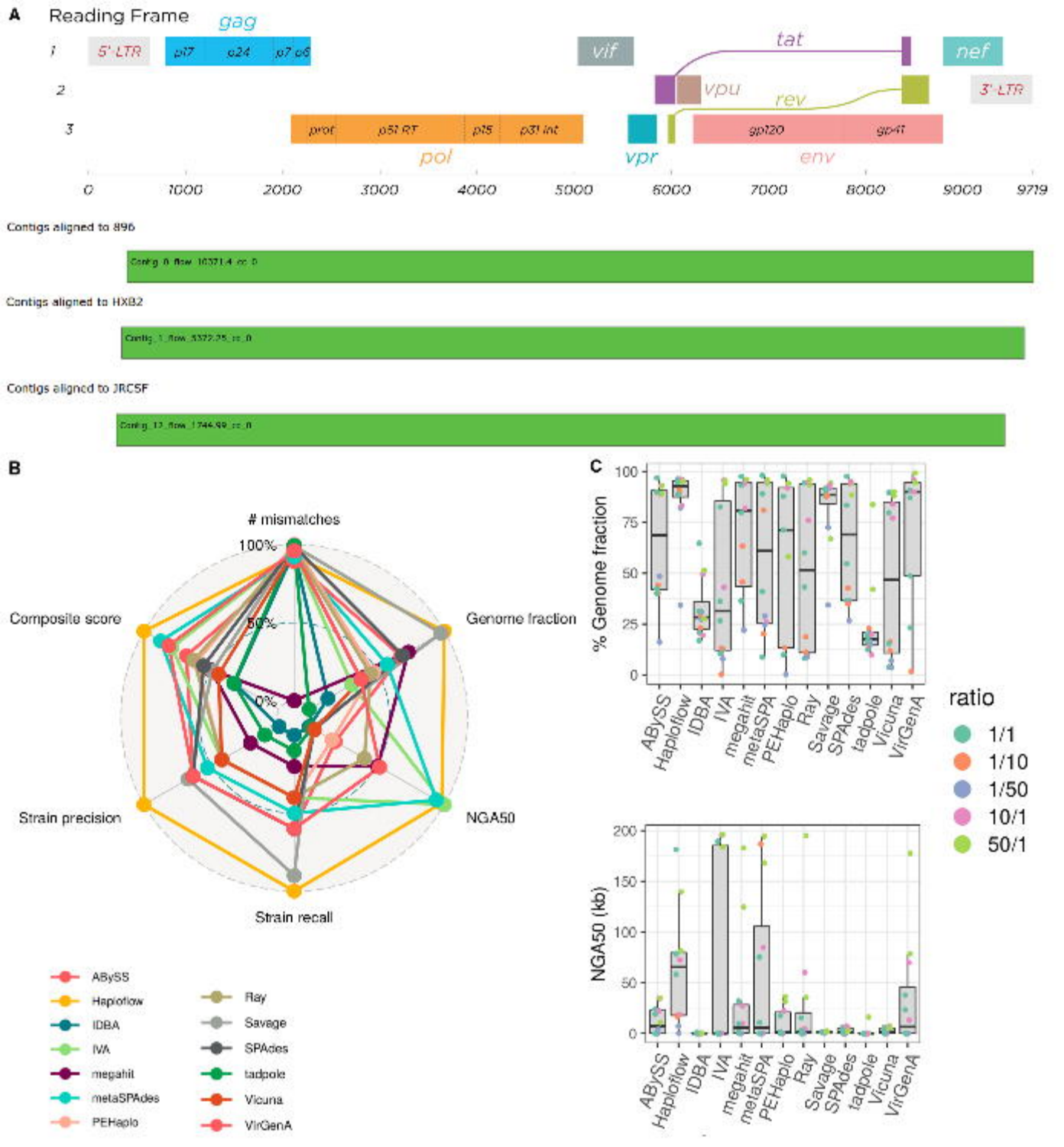
Algorithm 1: Fattest-path Dijkstra

Data: graph $G = (V, E)$ with edges $e = (v, w) \in E$, with $v, w \in V$, source $s \in V$, non-negative edge capacities $c(v, w)$ with $v, w \in V$

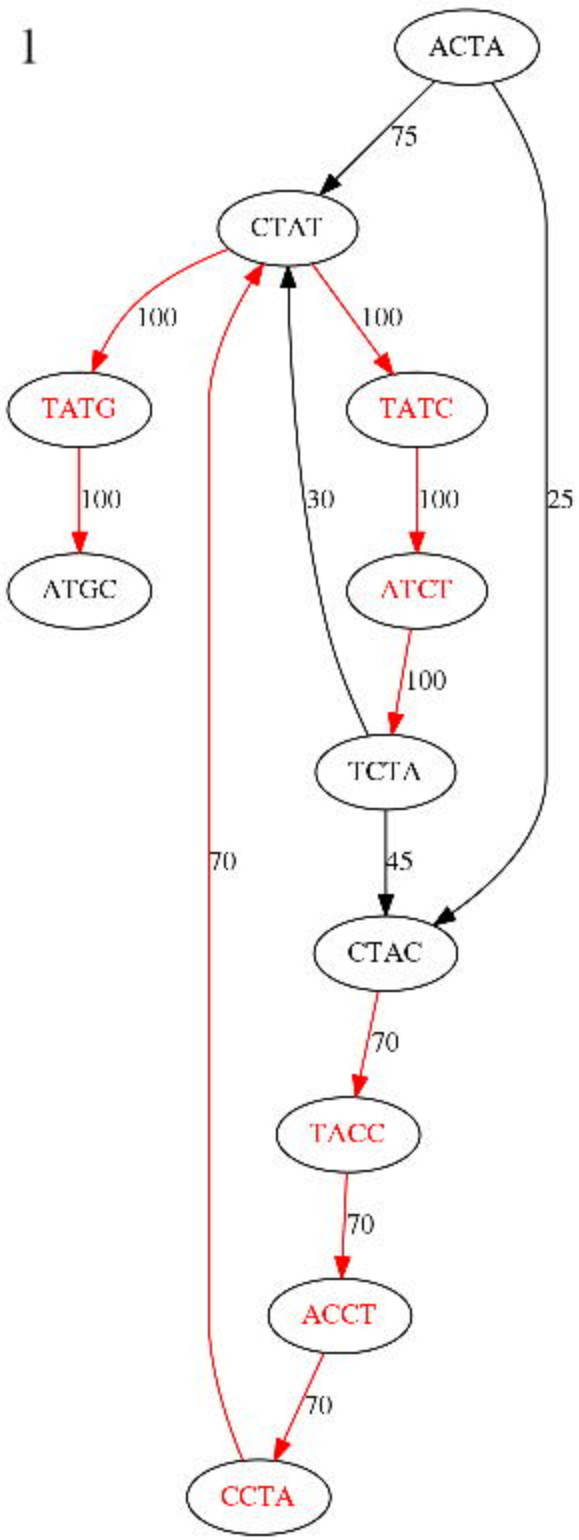
Result: Graph with edges labelled by their fatness, starting from source $s \in V$

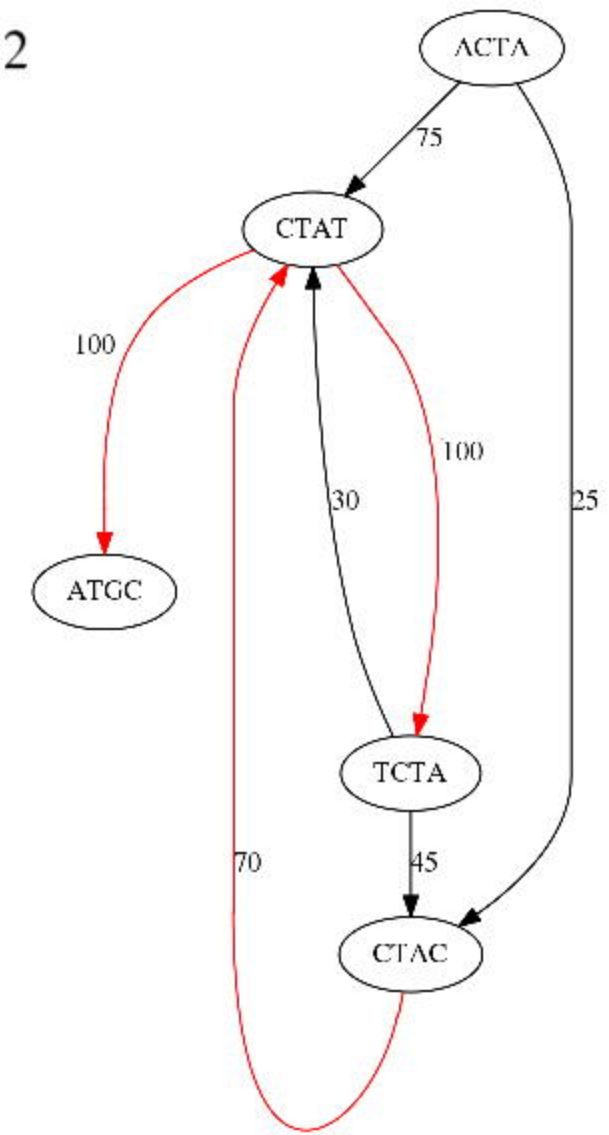
```
1  forall the v ∈ V − {s} do
2  |   v.fat = 0                                   set fatness to 0
3  |   s.dist = ∞                                  set distance to ∞
4  end
5  Q = priority_queue(V)          create priority queue keyed by v.dist
6  while !(Q.isEmpty()) do
7  |   u = argmax(Q, fat)                          select fattest vertex u in Q
8  |   del(u, Q)                                   remove u from Q
9  |   forall the v ∈ V s.t. (u, v) ∈ E do
10 |   |                                          Breadth-first search through G
11 |   |   if v.fat < min{u.fat, c(u, v)} then
12 |   |   |   v.fat = min{u.fat, c(u, v)}        update fatness
13 |   |   |   v.dist = u.dist + length(e)        set dist as distance to source
14 |   |   |   update Q with new dist values      update Q
15 |   |   |   u.pred = v                         set predecessor for backtracking
16 |   |   end
17 |   end
18 end
```


NC_045512.2
Germany/DUS_S2_minor/2020
C4655T
Germany/NRW-04/2020
C22323T
Germany/DUS_S2_major/2020
Germany/DUS_S3_major/2020
G2891A
Germany/DUS_S1_major/2020
Germany/DUS_S4/2020
Germany/DUS_S5_major/2020
Germany/NRW-02-1/2020
Germany/NRW-06/2020
G1440A
C11514T_C21575T
Germany/DUS_S5_minor/2020
A15139C
G15168A_C16954T
Germany/DUS_S3_minor/2020
T7717A_C13115T
Germany/DUS_S1_minor/2020
C22323T
A1440G_C8782T_C17747T_T28144C
USA/CA-CDPH-UC22/2020
G11467T_G12071A
USA/WIS_vero76_illumina_minor/2020
G11335T
USA/WIS_veroE6_illumina_minor/2020
C18110A_A19182G_C25658T
USA/WIS_veroSTAT-1KO_illumina_minor/2020
del.ATG518-520,del.AAGTCATTT686-694,C17373T
USA/WIS_P2b_vero76_illumina_minor/2020
del.ATT20298-20300
C10507T_C11897A_C19610T
C11454T
USA/WIS_P2a_vero76_illumina_minor/2020
USA/WI1/2020
C17373T
USA/WIS_veroSTAT-1KO_ONT/2020
USA/WIS_vero76_illumina_major/2020
USA/WIS_veroE6_ONT/2020
USA/WIS_vero76_ONT/2020
USA/WIS_swab_illumina/2020
USA/WIS_veroSTAT-1KO_illumina_major/2020
USA/WIS_veroE6_illumina_major/2020
USA/WIS_swab_ONT/2020
USA/WIS_P2b_vero76_illumina_major/2020
USA/WIS_P2a_vero76_illumina_major/2020
USA/WIS_P2b_vero76_ONT/2020
USA/WIS_P2a_vero76_ONT/2020
0
3
6
9
12


A
Reading Frame
gag
5' LTR
p17
p24
p7 p6
vif
tat
nef
vpu
rev
3' LTR
prot
p51 RT
p15
p31 int
pol
vpr
gp120
gp41
env
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
9719
Contigs aligned to 896
Contigs aligned to HXB2
Contigs aligned to JRCSF
B
mismatches
100%
50%
0%
Composite score
Genome fraction
Strain precision
NGA50
Strain recall
ABySS
Haploflow
IDBA
IVA
megahit
metaSPAdes
PEHaplo
Ray
Savage
SPAdes
tadpole
Vicuna
VirGenA
C
% Genome fraction
NGA50 (kb)
ratio
1/1
1/10
1/50
10/1
50/1

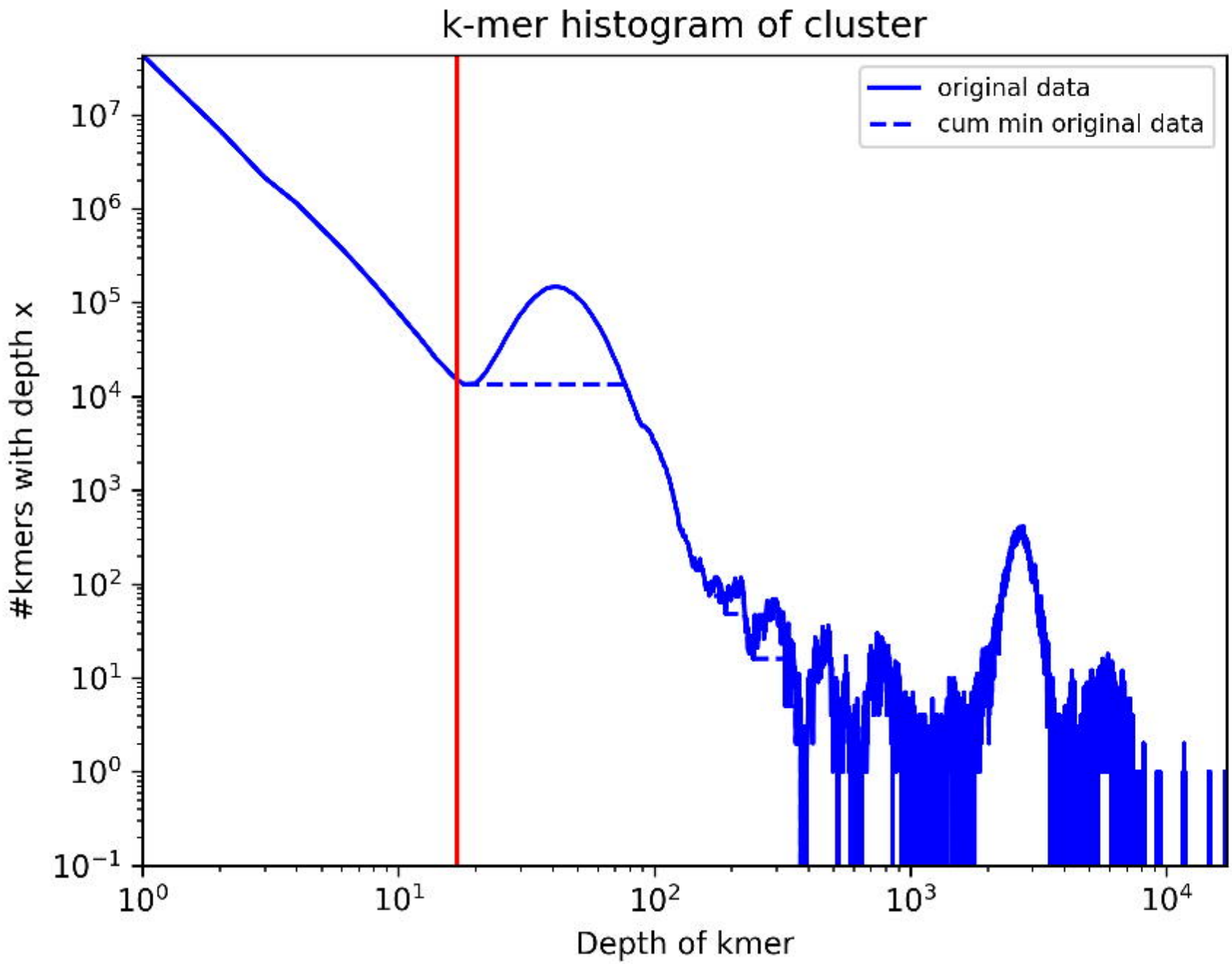
Software/ Dataset	**HIV 3 in silico mixture**		**HCMV in vitro mixture**		**Simulated virome**	
Metric	CPU *user* time (seconds)	Memory peak (GB)	Avg. CPU user time (seconds)	Avg. memory peak (GB)	CPU user time (seconds)	Memory peak (GB)
Haploflow	724	**0.009**	5,170	17.509	18,245	47.678
SAVAGE	110,208	102.938	75,518	17.658	-	-
PEHaplo	10,127	11.819	58,920	13.998	-	-
metaSPAdes	1,500	1.054	42,906	65.641	25,996	23.399
MEGAHIT	**250**	0.269	**2,910**	**0.754**	**9,690**	**2.148**



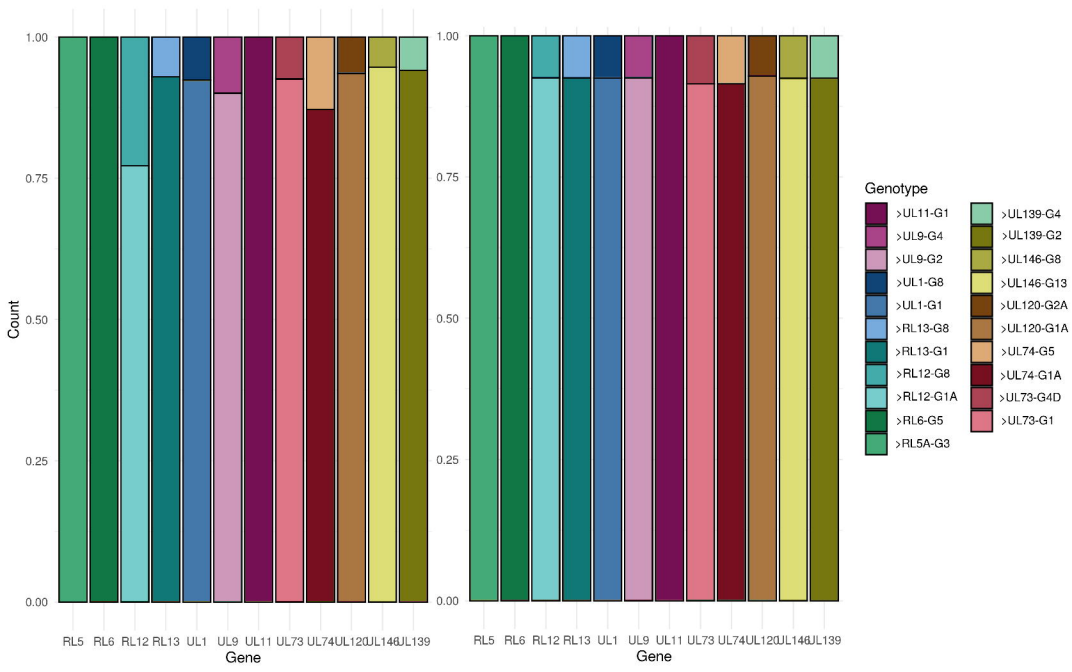


3

ACTATCTACCTATGC	45
ACTATCTA - - - - TGC	30
ACTACCTATCTATGC	25


k-mer histogram of cluster
original data
cum min original data
#kmers with depth x
Depth of kmer
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}

CliqueSNV
savage_bbnorm
Haploflow
MEGAHIT
metaSPAdes
PEHaplo
PEHaplo_bbnorm
PEHaplo_bbnorm_corr
PredictHaplo
SPAdes
Strain recall
100%
50%
0%
mismatches
Strain precision
Genome fraction
Composite score
NGA50


1.00
0.75
0.50
0.25
0.00
Count
RL5
RL6
RL12
RL13
UL1
UL9
UL11
UL73
UL74
UL120
UL146
UL139
Gene
Genotype
>UL11-G1
>UL9-G4
>UL9-G2
>UL1-G8
>UL1-G1
>RL13-G8
>RL13-G1
>RL12-G8
>RL12-G1A
>RL6-G5
>RL5A-G3
>UL139-G4
>UL139-G2
>UL146-G8
>UL146-G13
>UL120-G2A
>UL120-G1A
>UL74-G5
>UL74-G1A
>UL73-G4D
>UL73-G1

	Strain recall	**Strain precision**	**Compo-site score**	**Genome fraction (%)**	**Number of contigs**	**Mis-matches**	**Dupli-cation ratio**	**NGA50**
Haploflow	**3/3**	**3/3**	**9.66**	**93.36**	**3**	**9**	**1.001**	**9083**
metaSPAdes	0/3	0/3	2.78	33.70	9	62	**1.005**	1246
SPAdes	0/3	0/3	4.28	63.09	20	**1**	**1.021**	864
MEGAHIT	0/3	0/3	4.28	48.47	12	257	**1.084**	1701
PEHaplo *l=40*	0/3	0/9	5.86	**94.63**	40	1208	3.106	4305
PEHaplo[a]	0(1)/3	1/5	6.28	82.80	20	363	1.683	2774
PEHaplo[a] *correct*	0(2)/3	2/3	5.7	81.45	17	**24**	**1.044**	1773
SAVAGE *de novo**	-	-	-	-	-	-	-	-
SAVAGE *ref**	-	-	-	-	-	-	-	-
SAVAGE[a] *ref*	*2(3)/3*	*3/3*	6.74	*93.72*	18	*5*	*1.057*	1932
PredictHaplo	1/3	1/3	8.02	67.12	3	631	1.497	*9658*
QuasiRecomb**	0/3	0/3,507	-	67.12	3506	1,011,054	1748.6	*9649*
QuasiRecomb *conservative***	0/3	0/1,155	-	67.12	1154	411,576	575.67	*9654*
CliqueSNV	1/3	1/7	4.89	33.5	7	538	7.000	*9669*
GAEseq*	-	-	-	-	-	-	-	-
GAEseq[a]*	-	-	-	-	-	-	-	-

	Strain recall*	**Strain precision**	**Compo-site score**	**Genome fraction**	**Contigs**	**Mis-matches per 100kb**	**Duplication ratio**	**NGA50**
Haploflow	**10(3)/12**	**10/14**	**9.34**	**83.87± 10.37%**	20.50± 7.20	166.26± 122.97	1.20± 0.15	**62,560.42 ±35,233**
metaSPAdes	5(4)/12	5/12	8.18	58.57± 4.44%	17.42± 8.38	184.13± 337.14	**1.01± 0.01**	**60,008.25 ±37,089**
SPAdes	6(4)/12	6/12	5.22	65.52± 3.94%	85.17± 15.75	40.38± 29.84	1.05± 0.01	2,552.42 ±951
MEGAHIT	2(0)/12	2/23	4.71	68.01± 8.23%	324.83± 244.29	2254.09± 1901.7	1.92± 0.67	32,446.08 ±35,925
PEHaplo	4(3)/12	4/12	5.70	52.72± 18.07%	54.0± 78.17	**13.04± 11.68**	1.05± 0.07	10,960.1 ±6,602
tadpole	1(0)/12	1/12	3.15	24.47± 13.31%	39.92± 12.98	27.14± 50.67	**1.00± 0.00**	1,344.3± 3,292
ABySS	6(3)/12	6/12	6.41	64.88± 4.94%	20.92± 8.85	250.0± 85.39	1.05± 0.01	12,399.25 ±5,157
Ray	4(4)/12	4/12	5.90	51.39± 2.27%	16.67± 12.53	67.54± 63.39	1.07± 0.06	26,154.75 ±36,557
IDBA	0(0)/12	0/12	3.10	32.71± 9.52%	83.75± 13.24	104.46± 76.39	**1.03± 0.01**	154.67±178
Vicuna	4(1)/12	4/12	4.18	47.26± 0.93%	36.33± 7.79	104.05± 71.31	**1.02± 0.01**	2,657.67 ±704
IVA	4(3)/12	4/12	7.42	43.23± 16.2%	**11.92± 8.22**	121.61± 185.89	**1.02± 0.03**	**63,773.0 ±49,449**
VirGenA	*6(5)/12*	6/12	7.63	47.25± 1.35%	*5.67± 2.48*	102.41± 107.01	*1.01± 0.00*	33,324.58 ±30,049
SAVAGE	*9(5)/12*	9/17	4.61	*82.43± 15.69%*	283.17± 98.89	33.86± 28.37	1.46± 0.14	1,245.33 ± 349

	metaSPAdes	MEGAHIT	Haploflow
Genome fraction (lower abundance)	19.77±6.44%	30.00±12.6%	**76.67±12.24%**
Genome fraction (higher abundance)	**94.86±0.57%**	89.97±8.59%	91.62±5.38%
NGA50 (lower abundance)	0±0	0±0	**9,625±6,578**
NGA50 (higher abundance)	**149,712±48,744**	45,481±45,766	79,284±50,679

GI number	Scientific name	Within-cluster similarity max	Within-cluster similarity avg
118725053	Staphylococcus phage phiNM3	98.10%	93.32%
119443652	Staphylococcus phage phiPVL108	98.31%	94.59%
157102936	Staphylococcus prophage tp310-1	99.12%	95.47%
157102938	Staphylococcus prophage tp310-3	98.74%	95.72%
239507361	Staphylococcus phage phiPVL-CN125	98.31%	94.59%
257136356	Staphylococcus phage P954	96.12%	92.18%
29028667	Staphylococcus prophage phi 13	98.74%	95.73%
30043925	Staphylococcus prophage phiN315	98.10%	92.98%
41189515	Staphylococcus phage 77	96.12%	93.13%
9635165	Staphylococcus phage PVL	99.12%	95.25%
9635677	Staphylococcus prophage phiPV83	96.59%	93.86%

	SPAdes	MEGAHIT	Haploflow
Common strains	55.58	48.88	**62.85**
Common strains, coverage > 8	74.58	64.99	**89.36**
Total	**72.2**	68.6	66.6
Total, coverage > 8	93.07	87.55	**94.52**

Patient	Time points	Estimated low strain abundance (number of predicted strains)	Haploflow low strain abundance predictions (% and absolute value(s))	Genome fraction (portion of recovered genome) of strains vs. consensus sequence
RTR3	367 days	38.3% (2)	27.1% (43:16)	88.29% / 40.70%
	408 days	17.9% (2)	17.3% (167:35)	92.93% / 68.76%
RTR6	vitreous humour	9.4% (2)	12.7% (2576:373)	90.07% / 31.06%
	blood	17.3% (2)	18.4% (213:48)	99.07% / 92.73%
SCTR1	91 days	4.0% (2)	- (425)	99.69%
	126 days	22.9% (2)	21.2% (2429:653)	96.42% / 75.93%
	130 days	14.0% (2)	12.6% (313:45)	99.63% / 82.87%
	194 days	6.8% (2)	- (105)	99.78%
	224 days	37.8% (2)	26.8% (60:22)	92.73% / 79.36%
	231 days	25.4% (2)	23.6% (126:39)	92.09% / 37.63%
	244 days	- (1)	17.3% (81:17)	93.19%
	245 days	29.8% (2)	- (22)	92.86%
SCTR3	189 days	- % (1)	- (9)	98.74%
	272 days	20.5% (2)	24.3% (2781:893)	80.89% / 34.96%
	320 days	28.1% (2)	- (38)	98.46%
SCTR8	55 days	24.4% (2)	7.3% (178:14)	99.28% / 24.28%
	287 days	16.5% (2)	30.2%/8.3% (305:150:41)	93.86% / 58.5% / 52.33% (*)
SCTR11	88 days	15.6% (2)	16.3% (103:20)	99.07% / 85.76%
	192 days	11.6% (2)	11.2% (119:15)	91.75% / 78.20 %

SCTR17	21 days	34.7% (2)	22.1% (106:30)	94.81% / 30.96%
	28 days	30.6% (2)	17.6% (404:86)	98.39% / 17.04% (*)
	35 days	33.0% (2)	29.8%/9.5% (362:178:57)	99.46% / 28.08% / 22.61% (*)
	50 days	6.1% (2)	8.5% (4063:376)	97.26% / 94.18%
SCTR18	28 days	30.6% (3)	26.8% (101:37)	99.27% / 50.60% (*)
	35 days	30.3%/10.3% (3)	32.6%/10.2% (291:166:52)	80.79% / 81.88% / 48.85%
Summary strain detection		Total strains: 48	Recall: 91.7% (44/48) Precision: 93.6% (44/47)	

Position	Original base	Detected base	Detected by	Notes
518-520	ATG	---	Haploflow	
686-694	AAGTCATTT	---------	Haploflow	
1440	G	A	Haploflow, Lofreq	
2891	G	A	Haploflow	
4802	G	A	Lofreq	homopolymeric
7717	T	A	Haploflow, Lofreq	
10507	C	T	Haploflow, Lofreq	
11335	G	T	Haploflow, Lofreq	
11454	C	T	Haploflow, Lofreq	
11467	G	T	Haploflow, Lofreq	
11514	C	T	Haploflow, Lofreq	
11897	C	A	Haploflow, Lofreq	
12071	G	A	Haploflow	rare
13115	C	T	Haploflow, Lofreq	
15139	A	C	Haploflow, Lofreq	
15157	C	A	Lofreq	Strand bias, called by Haploflow then filtered
15168	G	A	Haploflow, Lofreq	homopolymeric
16954	C	T	Haploflow, Lofreq	
18110	C	A	Haploflow	rare
17373	C	T	Haploflow, Lofreq	
19182	A	G	Haploflow, Lofreq	
19610	C	T	Haploflow, Lofreq	
20298-20300	ATT	---	Haploflow	
21077	C	T	Lofreq	homopolymeric

21575	C	T	Haploflow	homopolymeric
22323	C	T	Haploflow, Lofreq	
25658	C	T	Haploflow, Lofreq	
29659	C	T	Lofreq	homopolymeric
29760	T	C	Lofreq	

Sample	# strains Haploflow	GISAID matches to strains
Oakland 5/19	2	hCoV-19/USA/LA-SR0328/2020 hCoV-19/USA/CA-CSMC67/2020
Oakland 5/19 (2)	2	hCoV-19/Poland/PL_P31/2020 hCoV-19/Beijing/DT-BJ01/2020
Oakland 5/28	2	hCoV-19/USA/CA-CSMC25/2020 hCoV-19/USA/CA-CSMC67/2020
Oakland 6/09	1	hCoV-19/France/IDF-10064DR/2020
Oakland 6/30	2	hCoV-19/USA/WA-UW-11903/2020 hCoV-19/France/IDF-10064DR/2020
Oakland 6/30 (2)	2	hCoV-19/USA/CA-CSMC25/2020 hCoV-19/USA/LA-SR0328/2020
Marin 7/1	2	hCoV-19/USA/VA-DCLS-1271/2020 hCoV-19/USA/WA-UW-11903/2020