Overview of Virus Metagenomic Classification Methods and Their Biological Applications

Sam Nooij1,2*, Dennis Schmitz1,2, Harry Vennema1, Annelies Kroneman1 and Marion P. G. Koopmans1,2

1 Emerging and Endemic Viruses, Centre for Infectious Disease Control, National Institute for Public Health and the Environment (RIVM), Bilthoven, Netherlands, 2 Viroscience Laboratory, Erasmus University Medical Centre, Rotterdam, Netherlands

Metagenomics poses opportunities for clinical and public health virology applications by offering a way to assess complete taxonomic composition of a clinical sample in an unbiased way. However, the techniques required are complicated and analysis standards have yet to develop. This, together with the wealth of different tools and workflows that have been proposed, poses a barrier for new users. We evaluated 49 published computational classification workflows for virus metagenomics in a literature review. To this end, we described the methods of existing workflows by breaking them up into five general steps and assessed their ease-of-use and validation experiments. Performance scores of previous benchmarks were summarized and correlations between methods and performance were investigated. We indicate the potential suitability of the different workflows for (1) time-constrained diagnostics, (2) surveillance and outbreak source tracing, (3) detection of remote homologies (discovery), and (4) biodiversity studies. We provide two decision trees for virologists to help select a workflow for medical or biodiversity studies, as well as directions for future developments in clinical viral metagenomics.

Keywords: pipeline, decision tree, software, use case, standardization, viral metagenomics

INTRODUCTION

Unbiased sequencing of nucleic acids from environmental samples has great potential for the discovery and identification of diverse microorganisms (Tang and Chiu, 2010; Chiu, 2013; Culligan et al., 2014; Pallen, 2014). We know this technique as metagenomics, or random, agnostic or shotgun high-throughput sequencing. In theory, metagenomics techniques enable the identification and genomic characterisation of all microorganisms present in a sample with a generic lab procedure (Wooley and Ye, 2009). The approach has gained popularity with the introduction of next-generation sequencing (NGS) methods that provide more data in less time at a lower cost than previous sequencing techniques. While initially mainly applied to the analysis of the bacterial diversity, modifications in sample preparation protocols allowed characterisation of
viral genomes as well. The fields of virus discovery and biodiversity characterisation have seized the opportunity to expand their knowledge (Cardenas and Tiedje, 2008; Tang and Chiu, 2010; Chiu, 2013; Pallen, 2014). There is interest among virology researchers to explore the use of metagenomics techniques, in particular as a catch-all for viruses that cannot be cultured (Yozwiak et al., 2012; Smits and Osterhaus, 2013; Byrd et al., 2014; Naccache et al., 2014; Pallen, 2014; Smits et al., 2015; Graf et al., 2016). Metagenomics can also be used to benefit patients with uncommon disease etiologies that otherwise require multiple targeted tests to resolve (Chiu, 2013; Pallen, 2014). However, implementation of metagenomics in the routine clinical and public health research still faces challenges, because clinical application requires standardized, validated wet-lab procedures, meeting requirements compatible with accreditation demands (Hall et al., 2015). Another barrier is the requirement of appropriate bioinformatics analysis of the datasets generated. Here, we review computational workflows for data analysis from a user perspective.

Translating NGS outputs into clinically or biologically relevant information requires robust classification of sequence reads—the classical “what is there?” question of metagenomics. With previous sequencing methods, sequences were typically classified by NCBI BLAST (Altschul et al., 1990) against the NCBI nt database (NCBI, 2017). With NGS, however, the analysis needs to handle much larger quantities of short (up to 300 bp) reads for which proper references are not always available and take into account possible sequencing errors made by the machine. Therefore, NGS needs specialized analysis methods. Many bioinformaticians have developed computational workflows to analyse viral metagenomes. Their publications describe a range of computer tools for taxonomic classification. Although these tools can be useful, selecting the appropriate workflow can be difficult, especially for the computationally less-experienced user (Posada-Cespedes et al., 2016; Rose et al., 2016).

A part of the metagenomics workflows has been tested and described in review articles (Bazinet and Cummings, 2012; Garcia-Etxebarria et al., 2014; Peabody et al., 2015; Sharma et al., 2015; Lindgreen et al., 2016; Posada-Cespedes et al., 2016; Rose et al., 2016; Sangwan et al., 2016; Tangerlini et al., 2016) and on websites of projects that collect, describe, compare and test metagenomics analysis tools (Henry et al., 2014; CAMI, 2016; ELIXIR, 2016). Some of these studies involve benchmark tests of a selection of tools, while others provide brief descriptions. Also, when a new pipeline is published the authors often compare it to its main competitors. Such tests are invaluable to assessing the performance and they help create insight into which tool is applicable to which type of study.

We present an overview and critical appraisal of available virus metagenomic classification tools and present guidelines for virologists to select a workflow suitable for their studies by (1) listing available methods, (2) describing how the methods work, (3) assessing how well these methods perform by summarizing previous benchmarks, and (4) listing for which purposes they can be used. To this end, we reviewed publications describing 49 different virus classification tools and workflows—collectively referred to as workflows—that have been published since 2010.

**METHODS**

We searched literature in PubMed and Google Scholar on classification methods for virus metagenomics data, using the terms “virus metagenomics” and “viral metagenomics.” The results were limited to publications between January 2010 and January 2017. We assessed the workflows with regard to technical characteristics: algorithms used, reference databases, and search strategy used; their user-friendliness: whether a graphical user interface is provided, whether results are visualized, approximate runtime, accepted data types, the type of computer that was used to test the software and the operating system, availability and licensing, and provision of a user manual. In addition, we extracted information that supports the validity of the workflow: tests by the developers, wet-lab experimental work and computational benchmarks, benchmark tests by other groups, whether and when the software had been updated as of 19 July 2017 and the number of citations in Google Scholar as of 28 March 2017 (Data Sheet 1; https://compare.cbs.dtu.dk/inventory#pipeline). We listed only benchmark results from *in silico* tests using simulated viral sequence reads, and only sensitivity, specificity and precision, because these were most often reported (Data Sheet 2). Sensitivity is defined as reads correctly annotated as viral—on the taxonomic level chosen in that benchmark—by the pipeline as a fraction of the total number of simulated viral reads (true positives / (true positives + false negatives)). Specificity as reads correctly annotated as non-viral by the pipeline as a fraction of the total number of simulated non-viral reads (true negatives / (true negatives + false positives)). And precision as the reads correctly annotated as viral by the pipeline as a fraction of all reads annotated as viral (true positives / (true positives + false positives)). Different publications have used different taxonomic levels for classification, from kingdom to species. We used all benchmark scores for our analyses (details are in Data Sheet 2). Correlations between performance (sensitivity, specificity, precision and runtime) and methodical factors (different analysis steps, search algorithms and reference databases) were calculated and visualized with R v3.3.2 (https://www.r-project.org/), using RStudio v1.0.136 (https://www.rstudio.com).

Next, based on our inventory, we grouped workflows by compiling two decision trees to help readers select a workflow applicable to their research. We defined “time-restrained diagnostics” as being able to detect viruses and classify to genus or species in under 5 h per sample. “Surveillance and outbreak tracing” refers to the ability of more specific identification to the subspecies-level (e.g., genotype). “Discovery” refers to the ability to detect remote homologs by using a reference database that covers a wide range of viral taxa combined with a sensitive search algorithm, i.e., amino acid (protein) alignment or composition search. For “biodiversity studies” we qualified all workflows that can classify different viruses (i.e., are not focused on a single species).

Figures were made with Microsoft PowerPoint and Visio 2010 (v14.0.7181.5000, 32-bit; Redmond, Washington, U.S.A.), R packages heatmap v1.0.8 and ggplot2 v2.2.1, and GNU Image Manipulation Program (GIMP; v2.8.22; https://www.gimp.org).
RESULTS AND WORKFLOW DESCRIPTIONS

Available Workflows
We found 56 publications describing the development and testing of 49 classification workflows, of which three were unavailable for download or online use and two were only available upon request (Table 1). Among these were 24 virus-specific workflows, while 25 were developed for broader use, such as classification of bacteria and archaea. The information of the unavailable workflows has been summarized, but they were not included in the decision trees. An overview of all publications, workflows and scoring criteria is available in Data Sheet 1 and on https://compare.cbs.dtu.dk/inventory#pipeline.

Metagenomics Classification Methods
The selected metagenomics classification workflows consist of up to five different steps: pre-process, filter, assembly, search and post-process (Figure 1A). Only three workflows (SRSA, Isakov et al., 2011, Exhaustive Iterative Assembly, Schürch et al., 2014, and VIP, Li et al., 2016) incorporated all of these steps. All workflows minimally included a “search” step (Figure 1B, Table 4), as this was an inclusion criterion. The order in which the steps are performed varies between workflows and in some workflows steps are performed multiple times. Workflows are often combinations of existing (open source) software, while sometimes, custom solutions are made.

Quality Control and Pre-processing
A major determinant for the success of a workflow is the quality of the input reads. Thus, the first step is to assess the data quality and exclude technical errors from further analysis. This may consist of several processes, depending on the sequencing method and demands such as sensitivity and time constraints. The pre-processing may include: removing adapter sequences, trimming low quality reads to a set quality score, removing low quality reads—defined by a low mean or median Phred score assigned by the sequencing machine—removing low complexity reads (nucleotide repeats), removing short reads, deduplication, matching paired-end reads (or removing unmated reads) and removing reads that contain Ns (unresolved nucleotides). The adapters, quality, paired-end reads and accuracy of repeats depend on the sequencing technology. Quality cutoffs for removal are chosen in a trade-off between sensitivity and time constraints: removing reads may result in not finding rare viruses, while having fewer reads to process will speed up the analysis. Twenty-four workflows include a pre-processing step, applying at least one of the components listed above (Figure 1B, Table 2). Other workflows require input of reads pre-processed elsewhere.

Filtering Non-target Reads
The second step is filtering of non-target, in this case non-viral, reads. Filtering theoretically speeds up subsequent database searches by reducing the number of queries, it helps reduce false positive results and prevents assembly of chimeraic virus-host sequences. However, with lenient homology cutoffs, too many reads may be identified as non-viral, resulting in loss of potential viral target reads. Choice of filtering method depends on the sample type and research goal. For example, with human clinical samples a complete human reference genome is often used, as is the case with SRSA (Isakov et al., 2011), RINS (Bhaduri et al., 2012), VirusHunter (Zhao et al., 2013), MePIC (Takeuchi et al., 2014), Ensemble Assembler (Deng et al., 2015), ViromeScan (Rampelli et al., 2016), and MetaShot (Fosso et al., 2017). Depending on the sample type and expected contaminants, this can be extended to filtering rRNA, mtRNA, mRNA, bacterial or fungal sequences or non-human host genomes. More thorough filtering is displayed by PathSeq (Kostic et al., 2011), SURPI (Naccache et al., 2014), Clinical PathoScope (Byrd et al., 2014), Exhaustive Iterative Assembly (Schürch et al., 2014), VIP (Li et al., 2016), Taxonomer (Flygare et al., 2016), and VirusSeeker (Zhao et al., 2017). PathSeq removes human reads in a series of filtering steps in an attempt to concentrate pathogen-derived data. Clinical PathoScope filters human genomic reads as well as human rRNA reads. Exhaustive Iterative Assembly removes reads from diverse animal species, depending on the sample, to remove non-pathogen reads for different samples. SURPI uses 29 databases to remove different non-targets. VIP includes filtering by first comparing to host and bacterial databases and then to viruses. It only removes reads that are more similar to non-viral references in an attempt to achieve high sensitivity for viruses and potentially reducing false positive results by removing non-viral reads. Taxonomer simultaneously matches reads against human, bacterial, fungal and viral references and attempts to classify all. This only works well on high-performance computing facilities that can handle many concurrent search actions on large data sets. VirusSeeker uses the complete NCBI nucleotide (nt) and non-redundant protein (nr) databases to classify all reads and then filter non-viral reads. Some workflows require a custom, user-provided database for filtering, providing more flexibility but requiring more user-input. This is seen in IMSA (Dimon et al., 2013), VirusHunter (Zhao et al., 2013), VirFind (Ho and Tzanetakis, 2014), and MetLab (Norling et al., 2016), although other workflows may accept custom references as well. In total, 22 workflows filter non-virus reads prior to further analysis (Figure 1B, Table 3). Popular filter tools are read mappers such as Bowtie (Langmead, 2010; Langmead and Salzberg, 2012) and BWA (Li and Durbin, 2009), while specialized software, such as Human Best Match Tagger (BMTagger, NCBI, 2011) or riboPicker (Schmieder, 2011), is less commonly used (Table 2).

Short Read Assembly
Prior to classification, the short reads may be assembled into longer contiguous sequences (contigs) and generate consensus sequences by mapping individual reads to these contigs. This helps filter out errors from individual reads, and reduce the amount of data for further analysis. This can be done by mapping reads to a reference, or through so-called de novo assembly by linking together reads based on, for instance, overlaps, frequencies and paired-end read information. In viral metagenomics approaches, de novo assembly is often the method of choice. Since viruses evolve so rapidly, suitable references are not always available. Furthermore, the short viral genomes
| Name | References | URL |
|------|------------|-----|
| CaPSID | Borozan et al., 2012 | https://github.com/capsid/capsid |
| ClassyFlu | Van der Auwera et al., 2014 | http://bioinf.uni-greifswald.de/ClassyFlu/query/init |
| Clinical PathoScope | Byrd et al., 2014 | https://sourceforge.net/p/pathoscope/wiki/clinical_pathoscope/ |
| DUDes | Pro et al., 2016 | http://sf.net/p/dudes |
| EnsembleAssembler | Deng et al., 2015 | https://github.com/xutaodeng/EnsembleAssembler |
| Exhaustive Iterative Assembly (Virus Discovery Pipeline) | Schürch et al., 2014 | – |
| FACS | Stranneheim et al., 2010 | https://github.com/SciLifeLab/facs |
| GenSeed-HMM | Alves et al., 2016 | https://sourceforge.net/projects/genseedhmm/ |
| Giant Virus Finder | Karepessi and Grothmuz, 2016 | http://pitgroup.org/giant-virus-finder |
| GOTTCHA | Freitas et al., 2015 | https://github.com/LANL-Bioinformatics/GOTTCHA |
| IMISA | Dimon et al., 2013 | https://sourceforge.net/projects/arron-imsa/?source=directory |
| IMISA-A | Cox et al., 2017 | https://github.com/JeremyCoxBMI/IMISA-A |
| Kraken | Wood and Salzberg, 2014 | https://github.com/derickwood/hrakamen |
| LMAT | Ames et al., 2013 | https://sourceforge.net/projects/lmat/ |
| MEGAN 4 | Huson et al., 2011 | http://ab.inf.uni-tuebingen.de/software/megan4/ |
| MEGAN Community Edition | Huson et al., 2016 | http://ab.inf.uni-tuebingen.de/data/software/megan6/download/welcome.html |
| MePIC | Takeuchi et al., 2014 | https://mepic.nih.go.jp/ |
| MetaShot | Fosso et al., 2017 | https://github.com/bfossom/MetaShot |
| metaViC | Modha, 2016 | https://github.com/sejmohameta/molviC |
| Metavir | Roux et al., 2011 | http://metavir-meb.univ-bpclermont.fr/ |
| Metavir 2 | Roux et al., 2014 | http://metavir-meb.univ-bpclermont.fr/ |
| MetLab | Norling et al., 2016 | https://github.com/norling/metlab |
| NBC | Rosen et al., 2011 | http://nbc.ece.drexel.edu/ |
| PathSeq | Kostic et al., 2011 | https://www.broadinstitute.org/software/pathseq/ |
| ProVIDE | Ghosh et al., 2011 | http://metagenomics.atc.ts.nim/binning/ProVIDE/ |
| QuasiQ | Poh et al., 2013 | http://www.statgen.nus.edu.sg/~software/quasq.html |
| READSCAN | Naeem et al., 2013 | http://cbrc.kaust.edu.sa/readscan/ |
| Rega Typing Tool | Kroneman et al., 2011; Pineda-Peña et al., 2013 | http://regatools.med.kuleuven.be/typing/v3/hiv/typingtool/ |
| RiemS | Scheuch et al., 2015 | https://www.fl.de/fileadmin/FLI/IVD/Microarray-Diagnostics/RIEMS.tar.gz |
| RINS | Bhaduri et al., 2012 | http://khabirlab.stanford.edu/tools-1/#tools |
| SMART | Cotten et al., 2014 | “Available upon request” |
| SRSA | Lee et al., 2016 | https://bitbucket.org/aiy/smart |
| SURPR | Isaakov et al., 2011 | “Available upon request” |
| Taxonomer | Flygare et al., 2016 | https://www.taxonomer.com/ |
| Taxy-Pro | Klingenberg et al., 2013 | http://gobics.de/TaxyPro/ |
| “Unknown pathogens from mixed clinical samples” | Gong et al., 2016 | – |
| vFam | Skewes-Cox et al., 2014 | http://derisilab.ucsf.edu/software/vFam/ |
| VIP | Li et al., 2016 | https://github.com/keylabdvoc/VIP |
| ViralFusionSeq | Li et al., 2013 | https://sourceforge.net/projects/viralfusionseq/ |
| Virana | Schelhorn et al., 2013 | https://github.com/schelhorn/virana |
| VirFind | Ho and Tzanetakis, 2014 | http://virfind.org/ |
| VIROME | Wommack et al., 2012 | http://virome.dbi.udel.edu/app/#view/home |
| ViromeScan | Rampelli et al., 2016 | https://sourceforge.net/projects/viromescan/ |
| VirusSorter | Roux et al., 2015 | https://github.com/simroux/VirusSorter |
| VirusFinder | Wang et al., 2013 | http://bioinf.mc.vanderbilt.edu/VirusFinder/ |
| VirusHunter | Zhao et al., 2013 | https://www.ibrigenetwork.org/#/profiles/9065559575893/innovations/103/ |
| VirusSeeker | Zhao et al., 2017 | https://wupathlabs.wustl.edu/virusseeker/ |
| VirusSeq | Chen et al., 2013 | http:// Odin.mdacc.tmc.edu/~xsu1/VirusSeq.html |
| ViVarSeq | Verboist et al., 2015 | http://sourceforge.net/projects/virotools/?source=directory |
| VMGAP | Lorenzi et al., 2011 | – |

*No website could be found, the workflow was unavailable.*
generally result in high sequencing coverage, at least for high-titre samples, facilitating de novo assembly. However, de novo assembly is liable to generate erroneous contigs by linking together reads containing technical errors, such as sequencing (base calling) errors and remaining adapter sequences. Another source of erroneous contigs may be when reads from different organisms in the same sample are similar, resulting in the formation of chimeras. Thus, de novo assembly of correct contigs benefits from strict quality control and pre-processing, filtering and taxonomic clustering—i.e., grouping reads according to their respective taxa before assembly. Assembly improvement by taxonomic clustering is exemplified in five workflows: Metavir (Roux et al., 2011), RINS (Bhaduri et al., 2012), VirusFinder (Wang et al., 2013), SURPI (in comprehensive mode) (Naccache et al., 2014), and VIP (Li et al., 2016). Two of the discussed workflows have multiple iterations of assembly and combine algorithms to improve overall assembly: Exhaustive Iterative Assembly (Schürch et al., 2014) and Ensemble Assembler (Deng et al., 2015). In total, 18 of the tools incorporate an assembly step (Figure 1B, Table 4). Some of the more commonly used assembly programs are Velvet (Zerbino and Birney, 2008), Trinity (Grabherr et al., 2011), Newbler (454 Life Sciences), and SPAdes (Bankevich et al., 2012) (Table 2).

**Database Searching**

In the search step, sequences (either reads or contigs) are matched to a reference database. Twenty-six of the workflows we found search with the well-known BLAST algorithms BLASTn or BLASTx (Altschul et al., 1990; Table 2). Other often-used programs are Bowtie (Langmead, 2010; Langmead and Salzberg, 2012), BWA (Li and Durbin, 2009), and Diamond (Buchfink et al., 2015). These programs rely on alignments to a reference database and report matched sequences with alignment scores. Bowtie and BWA, which are also popular programs for the filtering step, align nucleotide sequences exclusively. Diamond aligns amino acid sequences and BLAST can do either nucleotides or amino acids. As analysis time can be quite long for large datasets, algorithms have been developed to reduce this time by using alternatives to classical alignment. One approach is to match k-mers with a reference, as used in FACS (Stranneheim et al., 2010), LMAT (Ames et al., 2013), Kraken (Wood and Salzberg, 2014), Taxonomer (Flygare et al., 2016), and MetLab (Norling et al., 2016). Exact k-mer matching is generally faster than alignment, but requires a lot of computer memory. Another approach is to use probabilistic models of multiple sequence alignments, or profile hidden Markov models (HMMs). For HMM methods, protein domains are used, which allows the detection of more remote homology between query and reference. A popular HMM search program is HMMER (Mistry et al., 2013). ClassyFlu (Van der Auwera et al., 2014) and vFam (Skewes-Cox et al., 2014) rely exclusively on HMM searches, while VMGAP (Lorentzi et al., 2011), Metavir (Roux et al., 2011), VirSorter (Roux et al., 2015), and MetLab can also use HMMER.

All of these search methods are examples of similarity search—homology or alignment-based methods. The other search method is composition search, in which oligonucleotide frequencies or k-mer counts are matched to references. Composition search requires the program to be “trained” on reference data and it is not used much in viral genomics. Only two workflows discussed here use composition search: NBC (Rosen et al., 2011) and Metavir 2 (Roux et al., 2014), while Metavir 2 only uses it complementary to similarity search (Data Sheet 1).

All search methods rely on reference databases, such as NCBI GenBank (https://www.ncbi.nlm.nih.gov/genbank/), RefSeq (https://www.ncbi.nlm.nih.gov/refseq/), or BLAST nucleotide (nt) and non-redundant protein (nr) databases (ftp://ftp.ncbi.nlm.nih.gov/blast/db/). Thirty-four workflows use GenBank for their references, most of which select only reference sequences from organisms of interest (Table 2). GenBank has the benefits

**FIGURE 1** Generic pipeline scheme and breakdown of tools. (A) The process of classifying raw sequencing reads in 5 generic steps. (B) The steps that workflows use in gray. UPfMCS: “Unknown Pathogens from Mixed Clinical Samples”; MEGAN CE: MEGAN Community Edition.
| Workflow                                    | Search software     | Search database                                      | Filter software          | Filter database                                      | Assembly software | Analysis steps | All software used                                      |
|--------------------------------------------|---------------------|------------------------------------------------------|--------------------------|-----------------------------------------------------|-------------------|---------------|-------------------------------------------------------|
| CaPSID                                    | Novoaalign/Bowtie2 (any) | NCBI Genomes: GenBank viral, bacteria and fungi or custom | Same as search           | NCBI human                                          | –                 | S             | Python 2.7, MongoDB, OpenJDK, BioPython, pysam, Novoaalign, BioScope, JBrowse, Groovy-Grails |
| ClassyFlu                                 | HMMER               | NCBI Influenza Virus Resource                       | –                        | NCBI human                                          | –                 | S             | HMMER                                                |
| Clinical PathoSope                         | Bowtie2             | NCBI Genomes: viral, bacteria                       | Bowtie2                  | NCBI human                                          | –                 | SFS           | Python, Bowtie2                                       |
| DUOes                                     | Bowtie2             | DUOesDB                                              | Bowtie2                  | NCBI RefSeq: viral, bacteria                        | SOAPDenovo, ABysS, MetaVelvet, CAP3 | –             | Python, Bowtie2, SOAPDenovo, ABysS, MetaVelvet, CAP3, MegaBLAST, Bowtie2, VcsScreen |
| EnsembleAssembler                          | Megablast           | NCBI RefSeq: viral, bacteria                        | Bowtie2                  | NCBI human                                          | SOAPDenovo, ABysS, MetaVelvet, CAP3 | –             | Python, Bowtie2, SOAPDenovo, ABysS, MetaVelvet, CAP3, MegaBLAST, Bowtie2, VcsScreen |
| Exhaustive Iterative Assembly Virus       | BLAST, MAFFT, PhyML | NCBI BLAST nt + nr, NCBI viral proteins (per hit), Pfam, custom | BLASTn                   | NCBI human                                          | –                 | –             | Python, Newbler, GSMART, CAP3, BLAST, HMMER, MEME, MAST, Bioconductor, R |
| Discovery Pipeline                        | Bloom filter        | Custom                                               | –                        | –                                                   | –                 | S             | Perf, C                                               |
| Facs                                      | BLASTx              | Microviridae proteins (Floux et al., 2012)          | –                        | –                                                   | CAP3, Velvet, Newbler, SOAPDenovo, ABysS | –             | Perf, Python, BLAST                                  |
| Giant Virus Finder                        | BLAST               | NCBI BLAST nt, custom reference genomes             | –                        | –                                                   | –                 | S             | Perf, Python, BLAST                                  |
| GOTTCHA                                   | BWA (mem)           | Custom                                               | –                        | –                                                   | –                 | PS-pp         | Perf, BWA                                            |
| IMSA                                      | BLAST               | Custom                                               | Bowtie, BLAT, BLAST      | User-defined (human genome)                        | –                 | FS            | Python, Blast, Bowtie2, Blat                         |
| IMSA+A                                    | Bowtie2             | “The reference genome”                               | –                        | Oases, Velvet, Trinity                              | FAS               | –             | Python, IMSA, BLAST+, BLAT, Bowtie2, Oases, Velvet, Trinity |
| Kraken                                    | Kraken              | NCBI RefSeq: bacteria, NCBI Genomes: GenBank bacteria + archaea | –                        | –                                                   | –                 | S-pp          | C++, Perf                                             |
| LMAT                                      | Custom              | NCBI Genomes: GenBank bacteria                        | –                        | –                                                   | –                 | S             | gcc, Python, OpenMP, MPI                            |
| MEGAN CE                                  | DIAMOND             | NCBI BLAST nt                                       | –                        | –                                                   | –                 | S-pp          | Java, DIAMOND, InterPro2GO, SEED viewer, eggNOG viewer, KEGG |
| MEGANII                                   | BLAST               | NCBI BLAST nt + nr                                   | –                        | –                                                   | –                 | S             | Java                                                 |
| MePIC                                     | Megablast           | NCBI BLAST nt                                       | BWA                      | NCBI human                                          | –                 | PFS           | fastaq-mcf (ea-utils), BWA, Megablast                |
| MetaShot                                  | Bowtie2, TANGO      | NCBI GenBank: viral, bacteria fungi (from plantae), protista (from invertebrate), and NCBI RefSeq: viral, bacteria, fungi, Protista (from invertebrate) | STAR                     | NCBI human                                          | GRCh37/hg19, 2009 | –             | Python, Bash, FaQOxs, STAR, Bowtie2, TANGO         |

(Continued)
| Workflow | Search software | Search database | Filter software | Filter database | Assembly software | Analysis | All software used |
|----------|-----------------|-----------------|----------------|----------------|------------------|---------|------------------|
| metaViC  | DIAMOND         | NCBI RefSeq: Complete protein | RiboPicker     | –              | IDBA-UD, SPAdes  | PFSAS   | Bowtie2, DIAMOND, filter_fastq.pl, GARM, IDBA-UD, Kronotools, prinseq, QUAST, riboPicker, PFSAS, Tim Galtee |
| Metavir  | BLASTx, MUSCLE, PhyML, HMMER | NCBI RefSeq: viral, Pfam, NCBI BLAST nr | –              | –              | CAP3            | SASPh   | MUSCLE, CAP3, Gblocks, PhyML, Scriptree |
| Metavir 2| BLAST, custom, FastTree | NCBI RefSeq: viral, Pfam | –              | –              | –               | SPH     | Perl, Php, Javascript, C++, R, BLAST, FastTree, MetaGeneAnnotator, HMMScan, Uclust, jackhammer, RaphaelSVG, Cytoscape-web |
| MetLab   | Kraken, HMMER3  | NCBI RefSeq: bacteria, archaea, NCBI GenBank: viral, phage, vFams | Bowtie2 | “Host genome” (human) | SPAdes | P|F|Y|A|SS | Python (2.7) + libraries, GNU MFPF, Prinseq-Lite, Bowtie2, SAMTOOLS, SPAdes, Krona Tools, Kraken, FragGeneScan, HMMsearch, vFamParse |
| NBC      | Custom          | “Unique N-mer frequency profiles of 635 microbial genomes” | –              | –              | –               | S       | Python, C++ |
| PathSeq  | BLAST           | NCBI BLAST nr: viruses, gungi, NCBI Genomes: bacteria, NCBI BLAST nr | MAQ, Megablast, BLASTN | 1000 Genomes Project: female reference, Ensemble Homo sapiens cDNA, NCBI BLAST: human genome, transcriptome, NCBI human | Velvet  | P|F|F|A|S | Python, Java, C++, C, Hadoop, MAQ, Megablast, Blast, Velvet, RepeatMasker |
| ProViDE | BLASTx          | NCBI BLAST nr | –              | –              | –               | S       | Perl, Python |
| QuassQ   | Bowtie2         | “The reference genome” | –              | –              | –               | PS-pp   | Perl, Bowtie2 |
| READSCAN | SMALT          | ?               | SMALT          | ?              | –               | S       | Perl, SMALT, Makeflow |
| Rega Typing Tool v3 | BLAST, TreePuzzle | Custom | –              | –              | –               | SPH     | Php, Java, R, TreePuzzle |
| REIMS    | G3 Mapper, BLAST | NCBI BLAST nr, nr | –              | –              | Newbler         | PASAS(PP) | Bash, 454 genome sequencing software suite, Newbler, G3 mapper, #/# tools, BLAST, Emboss |
| RINS     | BLAT, BLAST     | Custom          | Bowtie         | Human genome   | Trinity         | SPFAS   | Blat, Bowtie, Trinity, BLAST |
| SUM      | BLAST           | NCBI GenBank entries of 2,000–500,000 bp long | Bowtie         | Human genome   | Trinity         | SPFAS   | Python, BLAST, QUASR, MEGAN, BWA, SPAdes, MUMmer |

(Continued)
| Workflow               | Search software | Search database                                                                 | Filter software | Filter database | Assembly software | Analysis steps | All software used |
|-----------------------|-----------------|---------------------------------------------------------------------------------|-----------------|-----------------|-------------------|----------------|--------------------|
| SMART                 | Custom          | NCBI GenBank: release v209                                                       |                 |                 |                   | S              | C+++, Ruby, Bash, Sickle, Google SparseHash, GNU parallel |
| SRSA                  | Megablast       | NCBI GenBank: CDS translations, PDB, SwissProt, PIR, PRF                        | BWA             |                 |                   | PFAS           | BWA, fastx_toolkit, BLAST, MegaBlast, Velvet             |
| SURPI                 | RAPSearch2, SNAP| "fast": NCBI RefSeq: bacteria genomic, NCBI BLAST nt + nr - viridae;           | SNAP            |                 |                   | PS             | Bash, Python, Perl, fastQValidator, Minomo, AbYSS, RAPSearch2, seqtk, SNAP, gt-sequniq, fastaq, cutadapt, prinseq-lite, diopiache |
| Taxonomer             | KAnalyze + custom k-mer matching | Uniref90 viruses                                                              | KAnalyze       |                 |                   | FFSS-pp        | Cython, KAnalyze                                      |
| Taxy-Pro              | CoMet           | Pfam + metagenomes                                                               |                 |                 |                   | S              | MATLAB, CoMet webserver                                |
| "Unknown pathogens from mixed clinical samples" | HMMER          | NCBI RefSeq: viral protein                                                       |                 |                 |                   | S              | CLC Genomics Workbench (7.5), Blast, Bowtie2, Clustal Omega, MEGAN, SimPlot |
| vFam                  | Bowtie2, RAPSearch2 (depending on model), MAFFT, ETE | "fast": VPR/IRD nucleotide DB; "sense": NCBI RefSeq: viral genomic, viral protein, NCBI GenBank viral neighbor genomes | Bowtie2         |                 |                   | S              | Shell, Python, Perl, PICARD, Bowtie2, vFAM, Velvet-Oases, RAPSearch2, ETE |
| VIP                   | Bowtie2, RAPSearch2 | "fast": VPR/IRD nucleotide DB; "sense": NCBI RefSeq: viral genomic, viral protein, NCBI GenBank viral neighbor genomes | Bowtie2         |                 |                   | S              | Shell, Python, Perl, PICARD, Bowtie2, vFAM, Velvet-Oases, RAPSearch2, ETE |
| ViralFusionSeq        | BWA, BLAST      | "Viral sequences and human decoy sequences"                                    |                 |                 |                   | PS             | Perl, BWA, BLAST                                       |
| Virana                | STAR, (BLAST, LASTZ) | NCBI RefSeq: "viruses," Repbase human endogenous retroviruses                    |                 |                 |                   | PS             | Python, STAR, BWA-mem, LASTZ, RazerS3, Jalview, Trinity, Oases |
| WrFind                | BLAST           | NCBI BLAST nt, NCBI RefSeq viral protein                                        | Bowtie2         |                 |                   | PFAS(S)        | Perl, fastx-toolkit, seq_crumbs, Bowtie2, Velvet, Blast, CAP3, Python |
| VIROME                | BLAST           | UniRef100, SEED, ACLAME, COG, QQ, KEGG, MGOL, CAMERA, UniVec                   | BLASTn, trNAscan-SE | "A tRNA subject database" |                   | PFSS           | Adobe Flex, MySQL, Blast, tRNA scan SE, MetaGene Annotator, CD-HI 454 |
| ViroScan              | Bowtie2         | NCBI Genomes: GenBank viral, in-house built reference databases                 | BMTagger        |                 |                   | SPFSS          | Bash, R, Perl, Java, Bowtie2, BmTagger, Picard         |
| VirSorter             | BLASTp, HMMER3  | Pfam, custom                                                                     |                 |                 |                   | PS             | Perl, HMMER3, MCL, MetaGeneAnnotator, MUSCLE, BLAST    |
| VirusFinder           | BLAST, BLAT, Bowtie2, BWA | RINS virus DB, or GIB-V                                                        | Bowtie2         |                 |                   | FSAS(S-pp)     | Perl, BLAST++, BLAT, Bowtie2, BWA, iCORN, CREST, GATK, SAMtools, SVDetect, Trinity |

(Continued)
TABLE 2 | Continued

| Workflow | Search database | Filter software | Analysis steps | Assembly software |
|----------|-----------------|-----------------|----------------|--------------------|
| VirusHunter | BLAST NCBI | Perl, MySQL, Blast, CD-HIT, RepeatMasker | All software used | VirusHunter |
| VirusSeeker | BLAST NCBI | Perl, SLURM, Blast, CD-HIT, RepeatMasker, Newbler, Phrap | VirusSeeker | BLAST, BWA-MEM, MegaBlast, BLASTx |
| VirusSeq | MOSAIK | Perl, MOSAIK | VirusSeq | MOSAIK |
| VMGAP | BLAST, HMMER | Perl, RepeatMasker, R, Forlan, Perl | VMGAP | BLAST, HMMER, BLASTx, TANANT |

A assembly, F, filter; P, pre-process; Ph, phylogeny; pp, post-process; S, search. –: not used/specified.

Post-processing
Classifications of the sequencing reads can be made by setting the parameters of the search algorithm beforehand to return a single annotation per sequence (cut-offs). Another option is to return multiple hits and then determine the relationship between the query sequence and a cluster of similar reference sequences. This process of finding the most likely or best supported taxonomic assignment among a set of references is called post-processing. Post-processing uses phylogenetic or other computational methods such as the lowest common ancestor (LCA) algorithm, as introduced by MEGAN (Huson et al., 2007). Six workflows use phylogeny to place sequences in a phylogenetic tree with homologous reference sequences and thereby classify them. This is especially useful for outbreak tracing to elucidate relationships between samples. Twelve workflows use other computational methods such as the LCA taxonomy-based algorithm to make more confident but less specific classifications (Data Sheet 1). In total, 18 workflows include post-processing (Figure 1B).

Usability and Validation
For broader acceptance and eventual application in a clinical setting, workflows need to be user-friendly and need to be validated. Usability of the workflows varied vastly. Some provide web-services with a graphical user-interface that work fast on any PC, whereas other workflows only work on one operating system, from a command line interface with no user manual. Processing time per sample ranges from minutes to several days (Table 3). Although web-services with a graphical user-interface are very easy to use, such a format requires uploading large GB-sized short read files to a distant server. The speed of upload and the constraint to work with one sample at a time may limit its usability. Diagnostic centers may also have concerns about the security of the data transferred, especially if patient-identifying reads and confidential metadata are included in the transfer. Validation of workflows ranged from high—i.e., tested by several groups, validated by wet-lab experiments, receiving frequent updates and used in many studies—to no evidence of validation (Table 4). Number of citations varied from 0 to 752, with six workflows having received more than 100 citations: MEGAN 4 (752), Kraken, (334), PathSeq (158), SURPI (128), of being a large, frequently updated database with many different organisms and annotation depends largely on the data providers. Other tools make use of virus-specific databases such as GIB-V (Hirahata et al., 2007) or ViPR (Pickett et al., 2012), which have the advantage of better annotation and curation at the expense of the number of included sequences. Also, protein databases like Pfam (Sonnhammer et al., 1998) and UniProt (UniProt, 2015) are used, which provide a broad range of sequences. Search at the protein level may allow for the detection of more remote homology, which may improve detection of divergent viruses, but non-translated genomic regions are left unused. A last group of workflows requires the user to provide a reference database file. This enables customization of the workflow to the user’s research question and requires more effort.
| Workflow   | PC Platform (Linux, Mac, Windows) | Graphical user-interface | Freely available | User manual | Runtime                |
|------------|----------------------------------|--------------------------|-----------------|-------------|------------------------|
| Taxonomer  | Any (webservice), or Linux, Mac OS | Yes/no (webservice/local installation) | Yes             | http://taxonomer.iobio.io/instructions.html | "Real-time, interactive" - < 10 min |
| Rega Typing Tool v3 | Any (webservice) | Yes (webservice) | Yes             | http://regatools.med.kuleuven.be/virus/typing/v3/typingtool/tutorial | 500 seqs in 5h |
| NBC        | Any (webservice) | Yes (webservice) | Yes             | http://nbc.ece.drexel.edu/tutorial.php | ± 21 h |
| MePIC      | Any (webservice) | Yes (webservice) | Use yes, download upon request | https://mepic.nih.go.jp/mepic/manual/ | 10h on 1 CPU, 6 min on 100 CPUs (Megablast only) |
| MetaVir 2  | Any (webservice) | Yes (webservice) | Use yes, download no | http://metavir-mib.iup-bp.dumont.fr/index.php?page=Tutorial | Hours-days |
| ViSorter   | Any (webservice) | Yes (webservice) | Yes             | https://github.com/simroux/ViSorter | Unknown |
| ClassyFlu  | Any (webservice), or Linux, Mac OS | Yes (webservice) | Yes             | htp://regatools.med.kuleuven.be/typing/v3/typingtool/tutorial | Unknown |
| VIROME     | Any (webservice) | Yes (webservice) | Use yes, download no | http://virome.db.u del.edu/, Tutorial videos | Unknown |
| ViFind     | Any (webservice) | Yes (webservice) | Use yes, download no | http://abi.inf.uni-tuebingen.de/software/megan6/download/manual.pdf | ±70h |
| CaPSID     | Linux, Mac OS | Yes               | Yes             | https://github.com/capsid/capsid/wiki | ±20min |
| MetLab     | Any                | Yes               | Yes             | https://github.com/norling/metlab/blob/master/INSTALL.md | <40min |
| MEGAN Community Edition | Any | Yes               | Yes             | http://abi.inf.uni-tuebingen.de/data/software/megan6/download/manual.pdf | ±5.5 h |
| MEGAN4     | Any                | Yes               | For academic use | http://abi.inf.uni-tuebingen.de/software/megan4/ | Unknown |
| Kraken     | Linux              | No (Illumina BaseSpace integration?) | Yes             | http://cdis.jhu.edu/software/kraken/ Manuel.html | ±1h |
| FACS       | Linux              | No                | Yes             | https://github.com/SciLifeLab/facs | "±20 times faster than BLAT/SSAHA2" |
| EnsembleAssembler | Linux | No                | Yes             | https://github.com/xutasdeng/EnsembleAssembler | <5min (on 8 CPU server) |
| ViromeScan | Linux, Mac OS | No                | Yes             | https://sourceforge.net/projects/viromescan/files/?source=navbar | 140 sequences/s/CPU |
| DUDEs      | Linux              | No                | Yes             | https://sourceforge.net/projects/duedes/files/README.md/download | 15-30 min |
| MetaShot   | Linux              | No                | Yes             | https://github.com/bfosso/MetaShot/blob/bfosso-patch-1/MetaShot%20User%20Guide.pdf | 2-3x slower than Kraken-MetaPhlAn2 |
| Clinical PathoScope | Any | No                | Yes             | https://sourceforge.net/p/pathoscope/wiki/clinical_pathoscope/ | <1h |
| READSCAN   | Linux              | No                | Yes             | http://www.cbrc.kaust.edu.sa/readscan/ , Supplied with download - in scripts | <27min on 16 CPU-HPC - 4h |
| Virana     | Linux, Mac OS      | No                | Yes             | https://github.com/schelhorn/virana | ±30 min/CPU |

(Continued)
| Workflow          | PC Platform (Linux, Mac, Windows) | Graphical user-interface | Freely available | User manual                              | Runtime                     |
|-------------------|----------------------------------|--------------------------|------------------|------------------------------------------|-----------------------------|
| SURPI             | Linux                            | No                       | Yes              | https://github.com/chiulab/surpi        | ±1 h (fast), ± 5 h (comprehensive) |
| RINS              | Linux                            | No                       | Yes              | Supplied with download(?)                | ±3h (2CPU), ±15 min (16CPU)  |
| IMSA              | Linux, Mac OS                    | No                       | Yes              | https://sourceforge.net/projects/aron-imsa/files/IMSA_UserManual_v2.pdf/download | ±4h ("2-5x slower than Kraken") |
| GOTTOCHA          | Linux, Mac OS                    | No                       | Yes              | https://github.com/LANL-Bioinformatics/GOTTOCHA | ±30 CPU hours               |
| Giant Virus Finder| Linux, Mac OS                    | No                       | Yes              | http://giltgroup.org/public/giant-virus-finder/latest/README |                           |
| VIP               | Linux (Ubuntu, Biinux)           | No                       | Yes              | https://github.com/keylabvipc/VIP       | < 2 d                       |
| VirusFinder       | Linux                            | No                       | Yes              | https://bioinfo.uth.edu/VirusFinder/VirusFinder-manual.pdf | 3 d                         |
| ViralFusionSeq    | Linux                            | No                       | Yes              | supplied with download                   | > 1 week                    |
| QuasQ             | Linux, Mac OS                    | No                       | Yes              | http://www.statgen.nus.edu.sg/S/fields/software/quasq.html | Unknown                     |
| IMSA+A            | Linux                            | No                       | Yes              | https://github.com/JeremyCoxBMI/IMSA-A/blob/master/IMSA%2BA_Detailed_Direction.pdf | Unknown                     |
| GenSeed-HMM       | Linux                            | No                       | Yes              | https://sourceforge.net/projects/genseedhmm/files | Unknown                     |
| VirVarSeq         | Linux, Mac OS                    | No                       | Yes              | https://sourceforge.net/projects/virtools/files/?source=navbar | Unknown                     |
| VirusFinder       | Linux                            | No                       | Yes              | https://www.wupathlabs.wustl.edu/virusfinder/usage/ | Unknown                     |
| vFam              | Linux, Mac OS                    | No                       | Yes              | –                                         | Unknown                     |
| metaVC            | Linux, Mac OS                    | No                       | Yes              | –                                         | Unknown                     |
| PathSeq           | Linux, cloud (Amazon EC2, Apache Hadoop) | No                       | Yes              | –                                         | Unknown                     |
| Taxy-Pro          | Any (webservice or MATLAB)       | No                       | Yes              | –                                         | "About three orders of magnitude faster than speed-optimized BLAST" |
| VirusSeq          | Linux, Mac OS                    | No                       | Yes              | –                                         | > 1 week                    |
| LIMAT             | Linux                            | No                       | Yes              | –                                         | 1.3 Mbp/s                   |
| RIEMS             | Linux                            | No                       | Yes              | –                                         | 10h (24 CPU-HPC)            |
| SMART             | Linux                            | No                       | Yes              | For academic use                         | < 10 min or ±2M reads/min (on HPC - 192 CPUs) |
| SLIM              | Linux, Mac OS                    | No                       | Upon request     | –                                         | hours of searching, hours for assembly per sample (almost 10x faster than BLAST) |

(Continued)
### Table 3 (Continued)

| Workflow | PC Platform (Linux, Mac, Windows) | Graphical user-interface | Freely available | User manual | Runtime | Notes |
|----------|----------------------------------|--------------------------|-----------------|-------------|---------|-------|
| ProVDE   | Linux (Ubuntu/Fedora)            | Yes (web service)        | ?               | No          | Interactive | No     |
| SPSA     | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| VirusHunter | Linux                        | Yes (web service)        | No              | No          | Unknown | No     |
| Metavir  | Any (webservice)                 | Yes (webservice)         | Superceded by newer version | No (only at JCVI) | Interactive | No     |
| VMGAP    | ?                                | No                       | No              | No          | Unknown | No     |

Workflows are sorted by: availability of a graphical user interface (yes-no), runtime (fast-slow), and availability (yes-limited-no). ?: No operating system specified; –: no user manual found.

Correlation between methods, runtime, and performance

For 17 workflows for which these data were available, we looked for correlations by plotting performance scores against the analysis steps included (Figure 3). Workflows that included a pre-processing or assembly step scored higher in sensitivity, specificity and precision. Contrastingly, workflows with post-processing on average scored lower on all measures. Pipelines that filter non-viral reads generally had a lower sensitivity and specificity and precision remained high.

Next, we visualized correlations between the used search algorithms and the runtime, and the performance scores (Figure 4). Different search algorithms had different performance scores on average. Similarity search methods had lower sensitivity, but higher specificity and precision than composition search. The use of nucleotide vs. amino acid search also affected performance. Amino acid sequences generally led to higher sensitivity and lower specificity and precision scores. Combining nucleotide sequences and amino acid sequences in the analysis seemed to provide the best results. Performance was generally higher for workflows that used more time.

Finally, we inventoried the overall runtime of 17 workflows (Table 5) and separated them based on the inclusion of analysis steps that seemed to affect runtime. This indicated that workflows

---

**Classification Performance**

Next, we summarized workflow performance by aggregating benchmark results on simulated viral data from different publications (Figure 2). Twenty-five workflows had been tested for sensitivity, of which 19 more than once. For some workflows, sensitivity varied between 0 and 100, while for others sensitivity was less variable or only single values were available.

For 10 workflows specificities, or true negative rates, were provided. Six workflows had only single scores, all above 75%. The other four had variable specificities between 2 and 95%.

Precision, or positive predictive value was available for sixteen workflows. Seven workflows had only one recorded precision score. Overall, scores were high (>75%), except for IMSA+A (9%), Kraken (34%), NBC (49%), and vFam (3-73%).

Runtimes had been determined or estimated for 36 workflows. Comparison of these outcomes is difficult as different input data were used (for instance varying file sizes, consisting of raw reads or assembled contigs), as well as different computing systems. Thus a crude categorisation was made dividing workflows into three groups that either process a file in a timeframe of minutes (12 workflows: CaPSID, Clinical PathoScope, DUDes, EnsembleAssembler, FACS, Kraken, LMAT, Metavir, MetLab, SMART, Taxonomer and Virana), or hours (19 workflows: Giant Virus Finder, GOTTCHA, IMSA, MEGAN, MePIC, MetaShot, Metavir 2, NBC, ProVDE, Readscan, Rega Typing Tool, RIEMS, RINS, SLIM, SURPI, Taxy-Pro, “Unknown pathogens from mixed clinical samples,” VIP and ViromeScan), or even days (5 workflows: Exhaustive Iterative Assembly, ViralFusionSeq, VirFind, VirusFinder and VirusSeq).

**Correlations Between Methods, Runtime, and Performance**

For 17 workflows for which these data were available, we looked for correlations by plotting performance scores against the analysis steps included (Figure 3). Workflows that included a pre-processing or assembly step scored higher in sensitivity, specificity and precision. Contrastingly, workflows with post-processing on average scored lower on all measures. Pipelines that filter non-viral reads generally had a lower sensitivity and specificity and precision remained high.

Next, we visualized correlations between the used search algorithms and the runtime, and the performance scores (Figure 4). Different search algorithms had different performance scores on average. Similarity search methods had lower sensitivity, but higher specificity and precision than composition search. The use of nucleotide vs. amino acid search also affected performance. Amino acid sequences generally led to higher sensitivity and lower specificity and precision scores. Combining nucleotide sequences and amino acid sequences in the analysis seemed to provide the best results. Performance was generally higher for workflows that used more time.

Finally, we inventoried the overall runtime of 17 workflows (Table 5) and separated them based on the inclusion of analysis steps that seemed to affect runtime. This indicated that workflows

---

**Classifiers**

| Workflow               | PC Platform (Linux, Mac, Windows) | Graphical user-interface | Freely available | User manual | Runtime | Notes |
|------------------------|----------------------------------|--------------------------|-----------------|-------------|---------|-------|
| ProVDE                 | Linux (Ubuntu/Fedora)            | Yes (web service)        | Yes             | Yes         | Interactive | No     |
| SPSA                   | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| VirusHunter            | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| Metavir                | Any (webservice)                 | Yes (webservice)         | Superceded by newer version | No (only at JCVI) | Interactive | No     |
| VMGAP                  | ?                                | No                       | No              | No          | Unknown | No     |

Workflows are sorted by: availability of a graphical user interface (yes-no), runtime (fast-slow), and availability (yes-limited-no). ?: No operating system specified; –: no user manual found.

**Correlations Between Methods, Runtime, and Performance**

For 17 workflows for which these data were available, we looked for correlations by plotting performance scores against the analysis steps included (Figure 3). Workflows that included a pre-processing or assembly step scored higher in sensitivity, specificity and precision. Contrastingly, workflows with post-processing on average scored lower on all measures. Pipelines that filter non-viral reads generally had a lower sensitivity and specificity and precision remained high.

Next, we visualized correlations between the used search algorithms and the runtime, and the performance scores (Figure 4). Different search algorithms had different performance scores on average. Similarity search methods had lower sensitivity, but higher specificity and precision than composition search. The use of nucleotide vs. amino acid search also affected performance. Amino acid sequences generally led to higher sensitivity and lower specificity and precision scores. Combining nucleotide sequences and amino acid sequences in the analysis seemed to provide the best results. Performance was generally higher for workflows that used more time.

Finally, we inventoried the overall runtime of 17 workflows (Table 5) and separated them based on the inclusion of analysis steps that seemed to affect runtime. This indicated that workflows

---

**Classifiers**

| Workflow               | PC Platform (Linux, Mac, Windows) | Graphical user-interface | Freely available | User manual | Runtime | Notes |
|------------------------|----------------------------------|--------------------------|-----------------|-------------|---------|-------|
| ProVDE                 | Linux (Ubuntu/Fedora)            | Yes (web service)        | Yes             | Yes         | Interactive | No     |
| SPSA                   | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| VirusHunter            | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| Metavir                | Any (webservice)                 | Yes (webservice)         | Superceded by newer version | No (only at JCVI) | Interactive | No     |
| VMGAP                  | ?                                | No                       | No              | No          | Unknown | No     |

Workflows are sorted by: availability of a graphical user interface (yes-no), runtime (fast-slow), and availability (yes-limited-no). ?: No operating system specified; –: no user manual found.

**Correlations Between Methods, Runtime, and Performance**

For 17 workflows for which these data were available, we looked for correlations by plotting performance scores against the analysis steps included (Figure 3). Workflows that included a pre-processing or assembly step scored higher in sensitivity, specificity and precision. Contrastingly, workflows with post-processing on average scored lower on all measures. Pipelines that filter non-viral reads generally had a lower sensitivity and specificity and precision remained high.

Next, we visualized correlations between the used search algorithms and the runtime, and the performance scores (Figure 4). Different search algorithms had different performance scores on average. Similarity search methods had lower sensitivity, but higher specificity and precision than composition search. The use of nucleotide vs. amino acid search also affected performance. Amino acid sequences generally led to higher sensitivity and lower specificity and precision scores. Combining nucleotide sequences and amino acid sequences in the analysis seemed to provide the best results. Performance was generally higher for workflows that used more time.

Finally, we inventoried the overall runtime of 17 workflows (Table 5) and separated them based on the inclusion of analysis steps that seemed to affect runtime. This indicated that workflows

---

**Classifiers**

| Workflow               | PC Platform (Linux, Mac, Windows) | Graphical user-interface | Freely available | User manual | Runtime | Notes |
|------------------------|----------------------------------|--------------------------|-----------------|-------------|---------|-------|
| ProVDE                 | Linux (Ubuntu/Fedora)            | Yes (web service)        | Yes             | Yes         | Interactive | No     |
| SPSA                   | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| VirusHunter            | Linux                            | Yes (web service)        | No              | No          | Unknown | No     |
| Metavir                | Any (webservice)                 | Yes (webservice)         | Superceded by newer version | No (only at JCVI) | Interactive | No     |
| VMGAP                  | ?                                | No                       | No              | No          | Unknown | No     |

Workflows are sorted by: availability of a graphical user interface (yes-no), runtime (fast-slow), and availability (yes-limited-no). ?: No operating system specified; –: no user manual found.
### TABLE 4 | Validation features of classification workflows.

| Workflow                  | Tested by                                      | Validation methods                      | Sensitivity (%) | Specificity (%) | Precision (%) | Updates (most recent update) | Citations (Google Scholar) |
|---------------------------|------------------------------------------------|-----------------------------------------|-----------------|-----------------|---------------|-------------------------------|----------------------------|
| Kraken                    | MetaShot, IMSA+A, Taxonomer, GOTTCHA, RIEMS, MetLab | –                                       | 67 (21)         | 92 (6)          | 97 (2)        | Yes (3-3-2016)                | 334                        |
| RINS                      | CaPSID, Virana, ReadScan, developers            | PCR + Sanger sequencing                 | 49 (16)         | 100 (4)         | 100 (4)       | Yes (10-1-2012)               | 51                         |
| CaPSID                    | Virana, developers                              | in vitro validation                     | 66 (8)          | 100 (4)         | 100 (4)       | Yes (2-6-2012)                | 26                         |
| MEGAN 4                   | MetLab, Bazinet and Cummings, 2012              | –                                       | x              | x               | x             | Yes (new version)             | 752                        |
| VirSorter                 | Developers                                     | Manual curation of prophages            | 62 (6)          | –               | 90 (6)        | Yes (15-2-2017)               | 34                         |
| Virana                    | Developers                                     | FISH, Southern blot                     | 67 (4)          | –               | 78 (4)        | Yes (1-6-2014)                | 9                          |
| vFam                      | Developers                                     | Compared to previous studies            | 33 (3)          | 99 (3)          | 34 (3)        | Yes (9-2-2014)                | 19                         |
| MEGAN Community Edition   | IMSA+A                                         | –                                       | x              | x               | x             | Yes (12-7-2017)               | 22                         |
| NBC                       | MetLab                                         | –                                       | 100 (1)         | 33 (5)          | 49 (1)        | Yes (28-7-2010)               | 125                        |
| SURPI                     | Taxonomer                                      | –                                       | 61 (3)          | –               | –             | Yes (5-6-2015)                | 128                        |
| PathSeq                   | Readscan, developers                           | –                                       | 51 (10)         | –               | –             | Yes (23-11-20164)             | 158                        |
| Metavir 2                 | ViromeScan                                     | –                                       | 82 (1)          | –               | –             | Yes (26-7-2016)               | 63                         |
| Clinical PathoScope       | RIEMS                                          | –                                       | 18 (13)         | –               | –             | Yes (21-6-2016)               | 21                         |
| ProVIDE                   | MetLab                                         | –                                       | 53 (1)          | 37 (5)          | 73 (1)        | No                            | 19                         |
| VirusSeq                  | –                                              | Serology, colorimetric in situ hybridization, immunohistochemistry | – | – | – | Yes (9-8-2013) | 50 |
| ViralFusionSeq            | –                                              | Sanger sequencing                      | –              | –               | –             | Yes (19-2-2017)               | 31                         |
| VIP                       | –                                              | "Independent confirmatory testing results" | – | – | – | Yes (21-2-2017) | 5 |
| VirusHunter               | –                                              | EM, serology (hemagglutination inhibition) | – | – | – | Unknown | 46 |
| SLIM                      | –                                              | RT-PCR                                  | –              | –               | –             | Yes³ (2017)                   | 27                         |
| "Unknown pathogens from mixed clinical samples" | –                                              | PCR, ELISA                              | – | – | – | Unknown | 1 |
| RIEMS                     | Developers                                     | –                                       | 91 (13)         | 100 (13)        | 100 (13)      | Yes (10-3-2015)               | 11                         |
| LMAT                      | Developers                                     | –                                       | 50 (6)          | –               | 93 (6)        | Yes (17-11-2016)              | 64                         |
| GOTTCHA                   | Developers                                     | –                                       | 71 (1)          | –               | –             | Yes (26-6-2017)               | 31                         |
| IMSA                      | Developers                                     | –                                       | 92 (4)          | –               | –             | Yes (17-4-2014)               | 10                         |
| READSCAN                  | Developers                                     | –                                       | 62 (15)         | –               | –             | No (16-9-2012)                | 30                         |
| FACS                      | Developers                                     | –                                       | 99 (2)          | 100 (2)         | –             | Yes (17-12-2015)              | 39                         |
| Taxonomer                 | Developers                                     | –                                       | 95 (4)          | 91 (1)          | –             | Yes (3-7-2017)                | 16                         |
| QuasQ                     | Developers                                     | –                                       | 96 (9)          | –               | 99 (9)        | Yes (10-7-2014)               | 5                          |
| ViromeScan                | Developers                                     | –                                       | 100 (1)         | –               | 100 (1)       | Yes (29-5-2017)               | 4                          |
| GenSeed-HMM               | Developers                                     | –                                       | 62 (4)          | –               | 82 (4)        | Yes (13-10-2016)              | 0                          |
| IMSA+A                    | Developers                                     | –                                       | 97 (8)          | –               | 81 (8)        | Yes (18-7-2017)               | 0                          |
| MetaShot                  | Developers                                     | –                                       | 98 (1)          | –               | 98 (1)        | Yes (22-6-2017)               | 0                          |
| SMART                     | –                                              | –                                       | –              | –               | –             | Yes (19-5-2016)               | 4                          |
| MetLab                    | –                                              | –                                       | –              | –               | –             | Yes (28-2-2017)               | 0                          |
| EnsembleAssembler         | –                                              | –                                       | –              | –               | –             | No (30-11-2014)               | 41                         |
| DUDes                     | –                                              | –                                       | –              | –               | –             | Yes (22-11-2016)              | 3                          |

(Continued)
that included pre-processing, filtering, and similarity search by alignment were more time-consuming than workflows that did not use these analysis steps.

Applications of Workflows

Based on the results of our inventory, decision trees were drafted to address the question of which workflow a virologist could use for medical and environmental studies (Figures 5, 6).

DISCUSSION

Based on available literature, 49 available virus metagenomics classification workflows were evaluated for their analysis methods and performance and guidelines are provided to select the proper workflow for particular purposes (Figures 5, 6). Only workflows that have been tested with viral data were included, thus leaving out a number of metagenomics workflows that had been tested only on bacterial data, which may be applicable to virus classification as well. Also note that our inclusion criteria leave out most phylogenetic analysis tools, which start from contigs or classifications.

The variety in methods is striking. Although each workflow is designed to provide taxonomic classification, the strategies employed to achieve this differ from simple one-step tools to analyses with five or more steps and creative combinations of algorithms. Clearly, the field has not yet adopted a standard method to facilitate comparison of classification results. Usability varied from a few remarkably user-friendly workflows with easy access online to many command-line programs, which are generally more difficult to use. Comparison of the results of the validation experiments is precarious. Every test is different and if the reader has different study goals than the writers, assessing classification performance is complex.

Due to the variable benchmark tests with different workflows, the data we looked at is inherently limited and heterogeneous. This has left confounding factors in the data, such as test data, references used, algorithms and computing platforms. These factors are the result of the intended use of the workflow, e.g., Clinical PathoScope was developed for clinical use and was not intended or validated for biodiversity studies. Also, benchmarks usually only take one type of data to simulate a particular use case. Therefore, not all benchmark scores are directly comparable and it is impossible to significantly determine correlations and draw firm conclusions.

We do highlight some general findings. For instance, when high sensitivity is required filtering steps should be minimized, as these might accidentally remove viral reads. Furthermore, the choice of search algorithms has an impact on sensitivity. High sensitivity may be required in characterization of environmental biodiversity (Tangherlini et al., 2016) and virus discovery. Additionally, for identification of novel variant viruses and virus discovery de novo assembly of genomes is beneficial. Discoveries typically are confirmed by secondary methods, thus reducing the impact in case of lower specificity. For example, RIEMS showed high sensitivity and applies de novo assembly. MetLab
combines de novo assembly with Kraken, which also displayed high sensitivity. When higher specificity is required, in medical settings for example, pre-processing and search methods with the appropriate references are recommended. RIEMS and MetLab are also examples of high-specificity workflows including pre-processing. Studies that require high precision benefit from pre-processing, filtering and assembly. High-precision methods are essential in variant calling analyses for the characterization of viral quasispecies diversity (Posada-Cespedes et al., 2016), and in medical settings for preventing wrong diagnoses. RINS performs pre-processing, filtering and assembly and scored high in precision tests, while Kraken also scored well in precision and with MetLab it can be combined with filtering and assembly as needed.

Clinicians and public health policymakers would be served by taxonomic output accompanied by reliability scores, as is possible with HMM-based search methods and phylogeny with bootstrapping, for example. Reliability scores could also be based on similarity to known pathogens and contig coverage. However, classification to a higher taxonomic rank (e.g., order) is more generally reliable, but less informative than a classification at a lower rank (e.g., species) (Randle-Boggis et al., 2016). Therefore, the use of reliability scores and the associated trade-offs need to be properly addressed per application.

Besides, medical applications may be better served by a functional rather than a taxonomic annotation. For example, a clinician would probably find more use in a report...
FIGURE 3 | Correlations between performance scores and analysis steps. Sensitivity, specificity and precision scores (in columns) for workflows that incorporated different analysis steps (in rows). Numbers at the bottom indicate number of benchmarks performed.

FIGURE 4 | Correlation between performance and search algorithm and runtime. Sensitivity, specificity and precision scores (in columns) for workflows that incorporated different search algorithms, using either nucleotide sequences, amino acid sequences or both, and workflows with different runtimes (rows). Numbers at the bottom indicate number of benchmarks performed.

of known pathogenicity markers than a report of species composition. Bacterial metagenomics analyses often include this, but it is hardly applied to virus metagenomics. Although valuable, functional annotation further complicates the analysis (Lindgreen et al., 2016).

Numerous challenges remain in analyzing viral metagenomes. First is the problem of sensitivity and false positive detections. Some viruses that exist in a patient may not be detected by sequencing, or viruses that are not present may be detected because of homology to other viruses, wrong annotation in databases or sample cross-contamination. These might both lead to wrong diagnoses. Second, viruses are notorious for their

Frontiers in Microbiology | www.frontiersin.org

April 2018 | Volume 9 | Article 749

Nooij et al. Virus Metagenomic Classification Workflows Overview
TABLE 5 | Correlation between runtime and method.

| Method                  | Minutes | Hours |
|-------------------------|---------|-------|
| Pre-process             | 1       | 6     |
| No pre-process          | 7       | 3     |
| Filter                  | 2       | 5     |
| No filter               | 6       | 4     |
| Assembly                | 2       | 3     |
| No assembly             | 6       | 6     |
| Nt sequences            | 6       | 6     |
| Aa sequences            | 1       | 1     |
| Nt + aa sequences       | 1       | 2     |
| Alignment               | 2       | 8     |
| Alignment + phylogeny   | 2       | 0     |
| Exact k-mer matching    | 3       | 0     |
| k-mer matching          | 1       | 0     |
| Composition search      | 0       | 1     |

Seventeen workflows, for which runtimes had been reported, were compared to find correlations between runtime and methods. Numbers indicate the number of workflows that process samples in a timeframe of either minutes or hours that use the method listed in the left column. Grayscale is proportional to the total number of scores per group, i.e., like a heatmap lower numbers are lighter and high numbers dark.

FIGURE 5 | Decision tree for selecting a virus metagenomics classification workflow for medical applications. Workflows are suitable for medical purposes when they can detect pathogenic viruses by classifying sequences to a genus level or further (e.g., species, genotype), or when they detect integration sites. Workflows can be applied to surveillance or outbreak tracing studies when very specific classification are made, i.e., genotypes, strains or lineages. A 1-day analysis corresponds to being able to analyse a sample within 5 h. Detection of novel variants is made possible by sensitive search methods, amino acid alignment or composition search, and a broad reference database of potential hits. Numbers indicate the number of workflows available on the corresponding branch of the tree.
provide directions for establishing best practices by pointing out which algorithms perform best in reproducible tests. For proper comparison, metadata such as sample preparation method and sequencing technology should always be included—and ideally standardized. Besides, true and false positive and negative results of synthetic tests have to be provided to compare between benchmarks.

Optimal strategies for particular goals should then be integrated in a user-friendly and flexible software framework that enables easy analysis and continuous benchmarking to evaluate current and new methods. The evaluation should include complete workflow comparisons and comparisons of individual analysis steps. For example, benchmarks should be done to assess the addition of a de novo assembly step to the workflow and measure the change in sensitivity, specificity, etc. Additionally, it remains interesting to know which assembler works best for specific use cases as has been tested by several groups (Treangen et al., 2013; Scholz et al., 2014; Smits et al., 2014; Vázquez-Castellanos et al., 2014; Deng et al., 2015). The flexible framework should then facilitate easy swapping of these steps, so that users can always use the best possible workflow. Finally, it is important to keep reference databases up-to-date by sharing new classified sequences, for instance by uploading to GenBank.

All these steps toward standardization benefit from implementation of a common way to report results, or minimum set of metadata, such as the MixS by the genomic standard consortium (Yilmaz et al., 2011). Currently several projects exist that aim to advance the field to wider acceptance by validating methods and sharing information, e.g., the CAMI challenge (http://cami-challenge.org/), OMICtools (Henry et al., 2014), and COMPARE (http://www.compare-europe.eu/). We anticipate steady development and validation of genomics techniques to enable clinical application and international collaborations in the near future.

**AUTHOR CONTRIBUTIONS**

AK and MK conceived the study. SN designed the experiments and carried out the research. AK, DS, and HV contributed to the design of the analyses. SN prepared the draft manuscript. All authors were involved in discussions on the manuscript and revision and have agreed to the final content.
ACKNOWLEDGMENTS

The authors would like to thank Matthew Cotten, Bas oude Munnink, David Nieuwenhuijse and My Phan from the Erasmus Medical Centre in Rotterdam for their comments during work discussions and critical review of the manuscript. Bas Dutilh and the bioinformatics group from the Utrecht University are thanked for their feedback on work presentations. Bram van Bunnik is thanked for making the table of workflow information available on the COMPARE website. Finally, Demelza Gudde is thanked for her feedback on the manuscript.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fmicb.2018.00749/full#supplementary-material
Hall, R. J., Draper, J. L., Nielsen, F. G. G., and Duttil, B. E. (2015). Beyond research: a primer for considerations on using viral metagenomics in the field and clinic. Front. Microbiol. 6:224. doi: 10.3389/fmicb.2015.00224

Henry, V. J., Bandrowski, A. E., Pepin, A. S., Gonzalez, B. J., and Desfeux, A. (2014). OMICTools: an informative directory for multi-omic data analysis. Database (Oxford). 2014:bau069. doi: 10.1093/database/bau069

Hirahata, M., Abe, T., Tanaka, N., Kuswana, Y., Shigemoto, Y., Miyazaki, S., et al. (2007). Genome Information Broker for Viruses (GIB-V): database for comparative analysis of virus genomes. Nucleic Acids Res 35, D339–D342. doi: 10.1093/nar/gkl1004

Ho, T., and Tzianitakis, I. E. (2014). Development of a virus detection and discovery pipeline using next generation sequencing. Virology 471–473, 54–60. doi: 10.1016/j.virol.2014.09.019

Huson, D. H., Auch, A. F., Qi, J., and Schuster, S. C. (2007). MEGAN analysis of metagenomic data. Genome Res. 17, 377–386. doi: 10.1101/gr.5969107

Huson, D. H., Beier, S., Flade, I., Górská, A., El-Hadidi, M., Mitra, S., et al. (2011). MEGAN community edition – interactive exploration and analysis of large-scale microbiome sequencing data. PLoS Comput. Biol. 12:e1004957. doi: 10.1371/journal.pcbi.1004957

Huson, D. H., Mitra, S., Ruscheweyh, H. I., Weber, N., and Schuster, S. C. (2011). Integrative analysis of environmental sequences using MEGAN4. Genome Res. 21, 1552–1560. doi: 10.1101/gr.120618.111

Isakov, O., Modai, S., and Shamron, N. (2011). Pathogen detection using RNA-seq data deep sequencing subtraction and assembly. Bioinformatics 27, 2027–2030. doi: 10.1093/bioinformatics/btr250

Kerépesi, C., and Grolmusz, V. (2016). Giant viruses of the Kutch Desert. [45x473]

Kostic, A. D., Ojesina, A. I., Pedamallu, C. S., Jung, J., Verhaak, R. G., Getz, G., et al. (2016). GIANT analysis of virus genomes. Nucleic Acids Res 35, D339–D342. doi: 10.1093/nar/gkl1004

Li, J. W., Wan, R., Yu, C. S., Co, N. N., Wong, N., and Chan, T. F. (2013). ViromeScan: a new tool for metagenomic viral community profiling. BMC Genomics 17:165. doi: 10.1186/1471-2164-14-355

Mistry, J., Finn, R. D., Eddy, S. R., Bateman, A., and Punta, M. (2013). Challenges in homology search: HMMER3 and convergent evolution of coiled-coil regions. Nucleic Acids Res. 41:e121. doi: 10.1093/nar/gkt263

Modha, S. (2016). semiVIC: Virus Metagenomics Pipeline for Unknown Host or in Absence of a Host Genome [Online]. Available online at: https://www.biostars.org

Naccache, S. N., Federman, S., Veeraraghavan, N., Zaharia, M., Lee, D., Samayoa, E., et al. (2014). A cloud-compatible bioinformatics pipeline for ultrarapid pathogen identification from next-generation sequencing of clinical samples. Genome Res. 24, 1180–1192. doi: 10.1101/gr.171934.113

Naern, R., Rashid, M., and Pain, A. (2013). READSCAN: a fast and scalable pathogen discovery program with accurate genome relative abundance estimation. Bioinformatics 29, 391–392. doi: 10.1093/bioinformatics/btt684

NCBI (2011). BMTagger: Best Match Tagger for Removing Human Reads from Metagenomics Datasets [Online]. Available online at: ftp://ftp.ncbi.nlm.nih.gov/pub/agarwal/bntagger/ (Accessed Feb 24, 2017).

NCBI (2017). NCBI Blast Databases [Online]. Available online at: ftp://ftp.ncbi.nlm.nih.gov/blast/db/ (Accessed Mar 7, 2017).

Nieuwenhuijse, D. F., and Koopmans, M. P. (2017). Metagenomic sequencing for surveillance of food- and waterborne viral diseases. Front. Microbiol. 8:230. doi: 10.3389/fmicb.2017.00230

Norling, M., Karlsson-Lindjö, O. E., Gourlé, H., Bongcam-Rudloff, E., and Hayer, I. (2016). MetLab: an in silico experimental design, simulation and analysis tool for viral metagenomics studies. PLoS ONE 11:e0160334. doi: 10.1371/journal.pone.0160334

Oulas, A., Pavloudi, C., Polymerakou, P., Pavlopoulost, G. A., Papalikoulanou, N., Kotoulas, G., et al. (2015). Metagenomics: tools and insights for analyzing next-generation sequencing data derived from biodiversity studies. Bioinform. Biol. Insights 9, 95–88. doi: 10.4137/BII.S12462

Pallen, M. J. (2014). Diagnostic metagenomics: potential applications to bacterial, viral and parasitic infections. Parasitology 141, 1856–1862. doi: 10.1017/S0031182014001304

Peabody, M. A., Van Rossum, T., Lo, R., and Brinkman, F. S. (2015). Evaluation of shotgun metagenomics sequence classification methods using in silico and in vitro simulated communities. BMC Bioinformatics 16:363. doi: 10.1186/s12859-015-0788-5

Posada-Cespedes, S., Seifert, D., and Beerenwinkel, N. (2016). Recent advances in comparative analysis of virus genomes. Virology 497, 239–257. doi: 10.1016/j.virusres.2016.09.016

Piro, V. C., Lindner, M. S., and Renard, B. Y. (2016). DUDes: a top-down taxonomic profiler for metagenomics. Bioinformatics 32, 2272–2280. doi: 10.1093/bioinformatics/btw150

Poh, W. T., Xia, E., Chiu-Inmanu, K., Wong, L. P., Cheng, A. Y., Malasit, P., et al. (2013). Virus quasispecies inference from 454 pyrosequencing. BMC Bioinformatics 14:355. doi: 10.1186/1471-2105-14-355

Rampelli, S., Soberini, M., Turroni, S., Quercia, S., Biagi, E., Brigidi, P., et al. (2016). ViromeScan: a new tool for metagenomic viral community profiling. BMC Genomics 17:165. doi: 10.1186/s12864-016-2446-3

Randle-Boggis, R. J., Helgason, T., Sapp, M., and Ashton, P. D. (2016). Evaluating techniques for metagenome annotation using simulated sequence data. FEMS Microbiol. Ecol. 92:fw695. doi: 10.1093/femsme/fw695

Rose, R., Constantinides, B., Tapinos, A., Robertson, D. L., and Prosperi, M. (2016). NBC: the Naive Bayes Classification tool webserver for taxonomic classification of metagenomic reads. Bioinformatics 22, 127–129. doi: 10.1093/bioinformatics/btw619
