bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Sunbeam: a pipeline for next-generation metagenomic sequencing experiments

Erik L. Clarke[1*], Louis J. Taylor[1*], Chunyu Zhao[2*], Andrew Connell[1], Frederic D. Bushman[1], Kyle Bittinger[2]

1 Department of Microbiology, University of Pennsylvania, Philadelphia, Pennsylvania 19104

2 Division of Gastroenterology, Hepatology and Nutrition, The Children's Hospital of Philadelphia, Philadelphia, Pennsylvania 19104

* These authors contributed equally to this work.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Abstract

Background: Metagenomic sequencing experiments require a number of preprocessing and analytical steps to interpret the microbial and genetic composition of biological samples. Such steps include quality control, adapter trimming, host decontamination, metagenomic classification, read assembly, and alignment to reference genomes.

Results: We present here an extensible and modular pipeline called Sunbeam that performs these steps in a consistent and reproducible way. It features a one-step installation and novel tools for eliminating artifactual sequences that may interfere with downstream analysis including Komplexity, a novel software tool to eliminate potentially problematic, low-complexity nucleotide sequences from metagenomic data. Another unique component of the Sunbeam pipeline is an easy-to-use extension framework that enables users to add custom processing or analysis steps directly to the Sunbeam workflow.

Conclusions: Sunbeam provides a foundation to build more in-depth analyses and to enable comparisons between disparate sequencing experiments by standardizing routine pre-processing and analytical steps. Sunbeam is written in Python using the Snakemake workflow management software and is freely available at github.com/sunbeam-labs/sunbeam under the GPLv3.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Background

Metagenomic shotgun sequencing involves isolating DNA from a microbial community of interest, then sequencing reads randomly drawn from the mixture. This is in contrast to marker gene sequencing (e.g., the 16S rRNA gene of bacteria), where specific target samples are amplified and sequenced. Metagenomic sequencing is revolutionizing research in microbiology and is beginning to be used in clinical diagnosis. However, an ongoing challenge is analyzing and interpreting the resulting large datasets in a standard and reliable fashion.

A common practice is to use Illumina sequencing technology to obtain a large number of short (100-250 base pair) reads from fragmented microbial DNA isolated from the sample of interest. In order to use these sequences to answer questions about the metagenome, there are a number of post-processing steps that must be carried out before analysis. Some of these steps are common to many other kinds of sequencing experiments, like quality control and sequencing adapter trimming, while others are unique to metagenomic sequencing, such as attributing each read to its species of origin.

A researcher has many tools at their disposal for accomplishing each post-processing step and will frequently encounter numerous parameters in each tool that change the resulting output and downstream analysis, sometimes radically. Varying parameters or tools between analyses makes it challenging to compare the results of different metagenomic sequencing experiments. Setting up and using a consistent workflow is important to ensure that the downstream analysis is reproducible. It is essential to document all the components and parameters used to produce the data and ensure that post-processing steps do not introduce confounding factors into the analysis.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

A metagenomic post-processing workflow should have the following qualities to maximize its utility and flexibility: it should be deployable on a wide range of computers; it should be easy to configure; it should provide robust error-handling and the ability to resume after interruptions; and it should be modular so that unnecessary steps can be skipped or ignored, or new procedures added. The ability to deploy the workflow on arbitrary computers ensures that the workflow can be repeated independent of a specific computing setup and removes any dependence on institution- or lab-specific resources. Similarly, the ability to record running parameters through the use of configuration files both allows flexibility and serves as documentation of what parameters were used. From an ergonomic standpoint, it is important that errors or interruptions in the workflow do not require it to be started from scratch, since modern sequencing experiments produce large amounts of data and processing steps are computationally time-consuming. The ability to resume after interruption saves both researcher time and money, especially in the case of cloud- or cluster-based deployment. Finally, not all of the post-processing steps in a workflow will be necessary for all experiments, and some experiments may require custom processing. In this case, the workflow should have an easy way to skip unnecessary steps while maintaining the ability to run those steps later if they become necessary. In addition, there should be a straightforward method for adding new steps into the workflow as needed.

Here, we introduce Sunbeam, an easily-deployable and configurable pipeline that produces a consistent set of post-processed files from Illumina sequencing experiments. Sunbeam is self-contained and installable on any modern Linux computer without any pre-existing dependencies or administrator privileges. It features robust error-handling, task resumption, and cluster operations resulting from its implementation in the Snakemake workflow language. Nearly all steps are configurable, with reasonable pre-specified defaults, allowing rapid deployment without extensive parameter tuning. Sunbeam is extensible using a simple mechanism that allows new workflow steps to be added at any point in the pipeline. In addition,

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Sunbeam features custom software that allow it to more robustly handle unusual sample types, especially low-quality or host-derived samples. These include custom-tuned host-derived read removal steps for any number of host or contaminant genomes, and Komplexity, a novel sequence complexity analysis program that rapidly and accurately removes low-complexity reads before downstream analysis. These tools allow Sunbeam to produce high-quality outputs regardless of the sample type.

Sunbeam is mostly implemented in Python and Rust and is licensed under the GPLv3. It is freely available at https://github.com/sunbeam-labs/sunbeam. Documentation is available at http://sunbeam.readthedocs.io.

Implementation

Installation

Sunbeam is self-contained—it manages and installs all of its own dependencies, and only requires Linux to run. Installation is performed by downloading the software from its repository and running "install.sh". Installation does not require administrator privileges.

Sunbeam architecture

Sunbeam is comprised of a number of steps that take specific files as inputs and produce other files as outputs. Because of its implementation in the Snakemake workflow language, the relationships between these steps can be described as a directed acyclic graph (DAG) where the dependencies between each step are known at runtime. This allows steps that do not rely on each other to operate independently on separate

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

processes or compute nodes. It also enables robust error-handling, as steps that fail or are interrupted do not cause other independent steps to stop. In addition, interrupted steps can be resumed without starting from scratch as long as the required inputs exist. Finally, individual outputs can be requested by the user and only the upstream steps that produce the required outputs will be run, allowing the user to skip or re-run any part of the pipeline in a modular fashion.

By default, Sunbeam performs the following operations on raw, demultiplexed Illumina sequencing reads in the following order:

1. Quality control: Adapter sequences are removed and bases are quality filtered using the Trimmomatic [1] and Cutadapt [2] software. Read pairs surviving quality filtering are kept.
2. Low-complexity masking: Sequence complexity in each read is assessed using Komplexity, a kmer-based complexity algorithm newly described here (below). Reads that fall below a user-customizable threshold are marked and removed.
3. Host read decontamination: Reads are mapped against a user-specified set of host or contaminant sequences using bwa [3]. Reads that map to any of these sequences within certain identity and length thresholds are marked and removed.

[[Figure 1: Flowchart/outline figure]]

A detailed workflow is shown in Figure 1. After this initial quality-control, optional downstream steps can be performed independently of each other. In the *classify* step, the decontaminated and quality-controlled reads are individually taxonomically classified using Kraken [4] and a user-supplied database. This provides an overview of the microbial composition of the samples. In the *assembly* step, reads from each

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

sample are assembled into contigs using MEGAHit [5]. Contigs above a pre-specified length are annotated for circularity and any open reading frames (ORFs) using Prodigal [6]. The contigs are then searched against any number of user-specified nucleotide or protein BLAST [7] databases, using both the entire contig and the putative genes identified by Prodigal. The results of these searches are summarized into reports for each sample's contigs. Finally, in the independent *mapping* step, quality-controlled reads are mapped using bwa to any number of user-specified reference genomes, and the resulting BAM files are sorted and indexed using samtools [8].

The standard outputs from Sunbeam include the reads from each step of the quality-control process, taxonomic assignments for each read, contigs built from each sample, taxonomic assignments and gene predictions for each contig, and alignment files of all reads to any number of reference genomes. Extensions can obtain the resulting files from any of these steps for further processing; for example, an extension could use the BAM alignments to auto-generate a coverage visualization, or classify quality-controlled reads using an alternative classifier such as Kaiju [9] (see this example at https://github.com/sunbeam-labs/sbx_kaiju). Each extension gains the same error-handling and interruption-protection as other steps in Sunbeam.

We have also incorporated an upgrade process and semantic versioning system in Sunbeam's development. Changes to Sunbeam that would be incompatible with previous versions of a user's config file will be marked with a 'major' change (i.e. version 1.0.0 to version 2.0.0) while bug fixes and non-breaking feature additions will be marked by increasing the minor or patch version number (i.e. 1.1.0 to 1.2.0). Users can upgrade Sunbeam in-place by running the included installation script. An included utility, 'sunbeam config update', updates user's existing config files between major version releases.

Komplexity architecture

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

We regularly encounter low-complexity sequences comprised of short nucleotide repeats that pose problems for downstream taxonomic assignment and assembly. To avoid these potential artifacts, we created a novel, fast read filter called Komplexity. Komplexity, implemented in the Rust programming language, is a stand-alone program designed to mask or remove these problematic, low-complexity nucleotide sequences. It scores sequence complexity by calculating the number of unique k-mers divided by the sequence length. Komplexity can either return this complexity score for the entire sequence or mask regions that fall below a score threshold. The k-mer length, window length, and complexity score cutoff are modifiable by the user, though default values are suggested. Komplexity accepts FASTA and FASTQ files as input and outputs either complexity scores or masked sequences in the input format. In its integration into the Sunbeam workflow, Komplexity assesses the total read complexity and removes reads that fall below the default threshold. Komplexity is available as a separate program at https://github.com/eclarke/komplexity.

Extensibility and deployment

Sunbeam is deployable on laptop computers, scientific computing clusters, and cloud computing platforms. Users can install the pipeline by downloading from our version-controlled software repository, then running an installation script provided in the download. After installation, users can check for new stable releases and upgrade by executing a single command. The pipeline comes with a test suite to verify that the system is installed correctly. During software development, this test suite is run automatically after each change is made to the repository, to ensure that the change did not introduce an error preventing the generation of correct output files.

The Sunbeam pipeline can be extended by users. Extensions take the form of supplementary *rules* written in the Snakemake format that define the additional steps to be executed. Optionally, two other files

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

may be provided: one listing additional software requirements, and another giving configuration options. Extensions can run in a separate software environment, allowing users to access additional software even if it has conflicting requirements to Sunbeam itself. To integrate these extensions, the user simply copies the files into Sunbeam's extension directory, where it is automatically integrated into the workflow when next run. The extension platform is tested as part of our integration test suite.

User extensions can be as simple or complex as desired and have minimal boilerplate: an extension with no additional dependencies can be as short as three lines of code. We provide an extension template on our GitHub page (https://github.com/sunbeam-labs/sbx_template) as well as a number of useful extensions (https://github.com/sunbeam-labs) that allow users to run alternate metagenomic read classifiers like Kaiju [9] or MetaPhlAn [10], visualize read mappings to reference genomes with IGV [11], and even format Sunbeam outputs for use with downstream analysis pipelines like Anvi'o [12].

Results and Discussion

Comparing low-complexity filtering program filtering and performance

Low complexity reads often cross-align between genomes, and commonly elude standard filters, so Sunbeam implements a new filter. A number of tools exist for filtering low-complexity nucleotide sequences. The gold standard, RepeatMasker [13], uses multiple approaches to identify and mask repeat or low complexity DNA sequences, including querying a database of repetitive DNA elements (either Repbase [14] or Dfam [15]). DUST [16] employs a unique algorithm which scores and masks nucleotide sequence windows that exceed a particular complexity score threshold such that no subsequence within the masked region has a

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

higher complexity score than the entire masked region. BBMask, developed by the Joint Genome Institute, masks sequences that fall below a threshold of k-mer Shannon diversity [17].

Many of these tools are not appropriate for use on metagenomic sequencing datasets. RepeatMasker uses databases of known repeat sequences to mask repetitive nucleotide sequences, but runs too slowly to be feasible for processing large datasets. Neither DUST nor RepeatMasker accept files in FASTQ format as input, requiring conversion to FASTA before processing. An option to filter reads falling below a certain complexity threshold is not available in DUST, RepeatMasker or BBMask (although filtering is available in the BBMask companion tool BBDuk). Finally, the memory footprint of BBMask scales with dataset size, requiring considerable resources to process large shotgun sequencing studies. Therefore, we designed Komplexity to mask or filter metagenomic reads as a rapid, scalable addition to the Sunbeam workflow that can also be installed and run independently. It accepts FASTQ files, can remove reads below specified threshold, and operates with a constant memory footprint.

To compare the performance of all the low-complexity-filtering tools discussed above, we used pIRS [18] to simulate Illumina reads from the human conserved CDS dataset [19] as well as human microsatellite records from the NCBI nucleotide database [20] (Supplemental File X) with the following parameters: average insert length of 170 nucleotides with a 5% standard deviation, read length of 100 nucleotides, and 5x coverage. To ensure compatibility with all programs, we converted the resulting files to FASTA format, then selected equal numbers of reads from both datasets for a total of approximately 1.1 million bases in the simulated dataset. We processed the reads using Komplexity, RepeatMasker, DUST and BBMask and used GNU Time (http://man7.org/linux/man-pages/man1/time.1.html) to measure peak memory usage and execution time for six replicates (Table 1). Komplexity and RepeatMasker mask a similar proportion of microsatellite nucleotides and none of the four tools masks a large proportion of coding nucleotides.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Komplexity runs faster and has a smaller memory footprint than other low-complexity filtering programs. The memory footprint of Komplexity and DUST are also relatively constant across datasets of different sizes.

[[Table 1: Komplexity speed comparison]]

To understand the extent to which different tools might synergize to mask a larger proportion of overall nucleotides, we visualized nucleotides from the microsatellite dataset masked by each tool or combinations of multiple tools using UpSetR [21] (Figure 2). Komplexity masks 78% of the nucleotides masked by any tool, and 96% excluding nucleotides masked by only RepeatMasker. This suggests that there would only be a marginal benefit to running other tools in series with Komplexity. The combination of Komplexity and host read decontamination mask or remove over 99% of the total simulated microsatellite reads, indicating that the workflow employed by Sunbeam is satisfactory in removing host-derived low-complexity nucleotide sequences.

[[Figure 2: Komplexity comparison of host read filtering]]

Comparison with other pipelines

There are a number of published and unpublished metagenomic pipelines available that offer subsets of the functionality provided by Sunbeam. These generally have their own specific use-case and target audience, and the steps they perform are well-tuned to those purposes. Here we review a number of the most developed pipelines and compare the pros and cons of these pipelines with Sunbeam.

Pipelines comparable to Sunbeam include SURPI (Sequence-based Ultra-Rapid Pathogen Identification) [22], KneadData [23], EDGE (Empowering the Development of Genomics Expertise) [24] and ATLAS (Automatic Tool for Local Assembly Structures) [25]. We show specific features and tools included in each pipeline in Table 2. Some pipelines address specific aspects or applications of metagenomic analysis--for

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

example, SURPI has been used to detect pathogens rapidly in clinical settings using shotgun sequencing, KneadData focuses on quality control to prepare metagenomic data, and EDGE will provide a web-based graphical user interface to make metagenomic analysis feasible for researchers without extensive computational expertise.

[[Table 2: Feature comparison for metagenomic pipelines]]

Sunbeam is designed to be robust, modular and extensible. We ensure robustness both by maintaining separate development and stable branches on GitHub and by employing continuous, full-pipeline integration testing using Circle CI. Sunbeam fully handles its own dependencies through the Conda environment manager (c) 2017 Continuum Analytics, Inc. (dba Anaconda, Inc.). https://www.anaconda.com), which alleviates any requirement for administrator privileges and protects against software version conflicts. ATLAS also uses Conda for dependency management and most other pipelines fully or partially install dependencies as part of the install process: SURPI handles dependencies using bash, the downloadable version of EDGE uses bash and custom archive files and KneadData handles some dependencies using the Python package manager, pip.

Analyses in Sunbeam are inherently modular, robust, and customizable as Sunbeam uses Snakemake workflows to manage data processing steps. Other pipelines also include complete or partial modularity including ATLAS, which also uses Snakemake, EDGE which allows for inclusion or exclusion of processing steps using the GUI, and SURPI, which can be run in fast or comprehensive mode depending on whether results are required in a clinically relevant timeframe.

We designed Sunbeam to be scalable and parallelizable; thanks to its implementation in Snakemake, Sunbeam rules have built-in parallelization and can individually specify the compute resources they require. For instance, if some tasks require fewer than the total available CPU cores on the machine, they can be run

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

in parallel. Likewise, each individual rule can be run on separate compute nodes on a cluster for massive parallelization.

As we understand that most users will want to perform different downstream analyses, Sunbeam includes a Snakemake-based framework for extensions. While many output files (e.g. quality controlled reads) from Sunbeam and other pipelines including ATLAS, KneadData and EDGE are made available for downstream processing using other metagenomic analysis and visualization software like Anvi'o [9], MetaPhlAn [10], and Pavian [26], Sunbeam's unique extension framework allows users to build their downstream analyses directly into the Sunbeam pipeline. This extension framework promotes reproducible analyses from start to finish and greatly simplifies performing the same type of analysis on multiple datasets. Extension templates, as well as a number of pre-built extensions, are available on our GitHub page (https://github.com/sunbeam-labs).

The Sunbeam extension sbx_report (https://github.com/sunbeam-labs/sbx_report) exemplifies the ease of incorporating custom analysis scripts into the Sunbeam workflow. The extension generates a HTML report from an R markdown script including useful information about read quality, sequencing depth and taxonomic classification. Figure 3 shows some of the plots output by sbx_report generated after analyzing a subset of a previously published dataset of healthy humans or individuals with Crohn's disease [27]. Our goal is to make extending Sunbeam as easy as possible--the sbx_report extension (excluding the R script) totals only 12 lines of code in two files.

[[Figure 3: Sbx_reports extension output]]

Conclusions

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Here we introduce Sunbeam, a Snakemake-based pipeline for analyzing shotgun metagenomic data with a focus on reproducible analysis, ease of deployment and use, and compare Sunbeam with other pipelines for metagenomic analysis. We also present Komplexity, a tool for rapidly filtering and masking low-complexity sequences from metagenomic sequence data, and show its superior performance in comparison with other tools for masking human microsatellite repeat sequences. Sunbeam's scalability, customizability, and ease of deployment simplify the processing of metagenomic sequence data, while its extension framework and thorough filtering and quality control enable robust and reproducible analyses. As a case study for Sunbeam's ease-of-use and robust deployability, we used Sunbeam at a metagenomics workshop at the University of Pennsylvania in the summer of 2017. All 25 participants successfully installed and ran the full pipeline on sample datasets, which emphasizes the ease of deploying and using Sunbeam.

Availability and requirements

Sunbeam is licensed under the GPLv3 and is freely available at https://github.com/sunbeam-labs/sunbeam. For detailed documentation and install instructions, see http://sunbeam.readthedocs.io. Sunbeam automatically installs and manages all dependencies and can be installed on any modern Linux computer.

List of abbreviations

ATLAS: Automatic Tool for Local Assembly Structures; BAM: binary alignment map; BLAST: Basic Local Alignment Search Tool; DAG: directed acyclic graph; DNA: deoxyribonucleic acid; EDGE: Empowering the Development of Genomics Expertise; GUI: graphical user interface; GPL: (GNU) General Public License;

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

ORF(s): open reading frame(s); SDUST: Symmetric DUST; SURPI: Sequence-based Ultra-Rapid Pathogen Identification

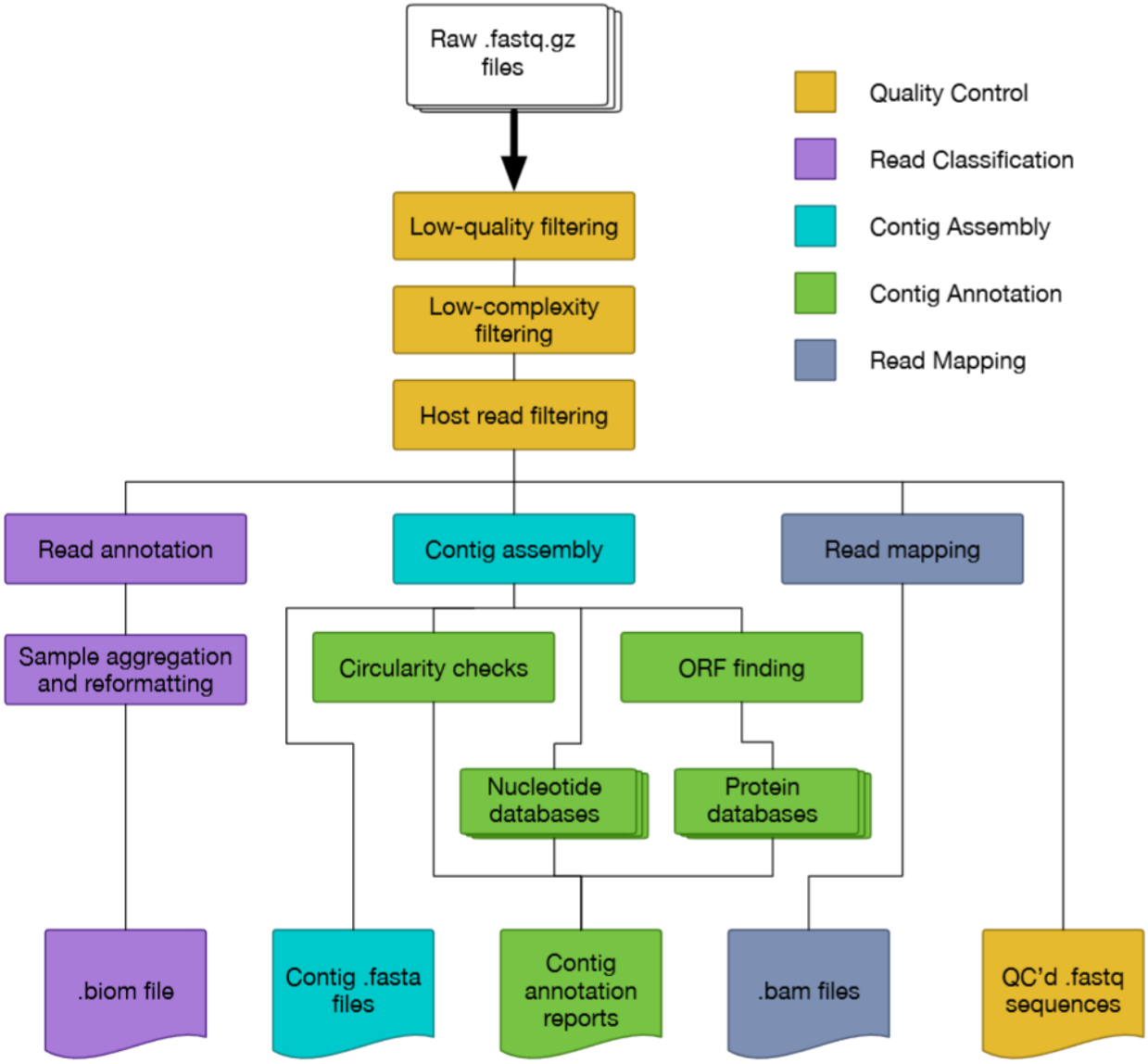
Acknowledgements

Thanks to members of the Bushman lab, Penn-CHOP Microbiome Center, and Penn Bioinformatics Code Review communities for helpful suggestions, discussions, and beta testing. This work was supported by the NIH grants U01HL112712 (Site-Specific Genomic Research in Alpha-1 Antitrypsin Deficiency and Sarcoidosis (GRADS) Study) and R01HL113252, and received assistance from the Penn Center for AIDS Research (P30AI045008), T32 Training Grant (T32-AI-007324, LJT), and the Penn-CHOP Microbiome Program.

Figures and Legends

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

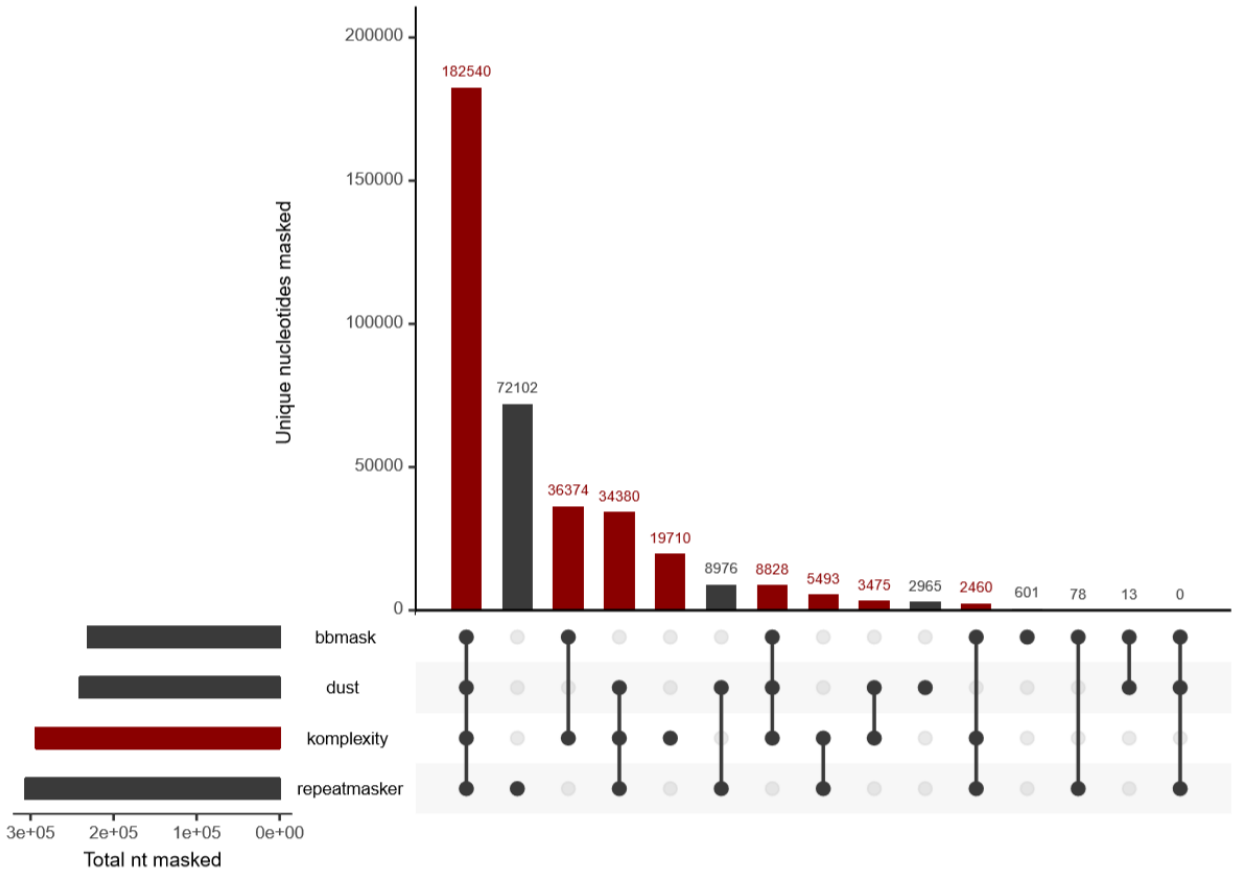
Figure 1



Flowchart of inputs, processes and outputs using Sunbeam.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

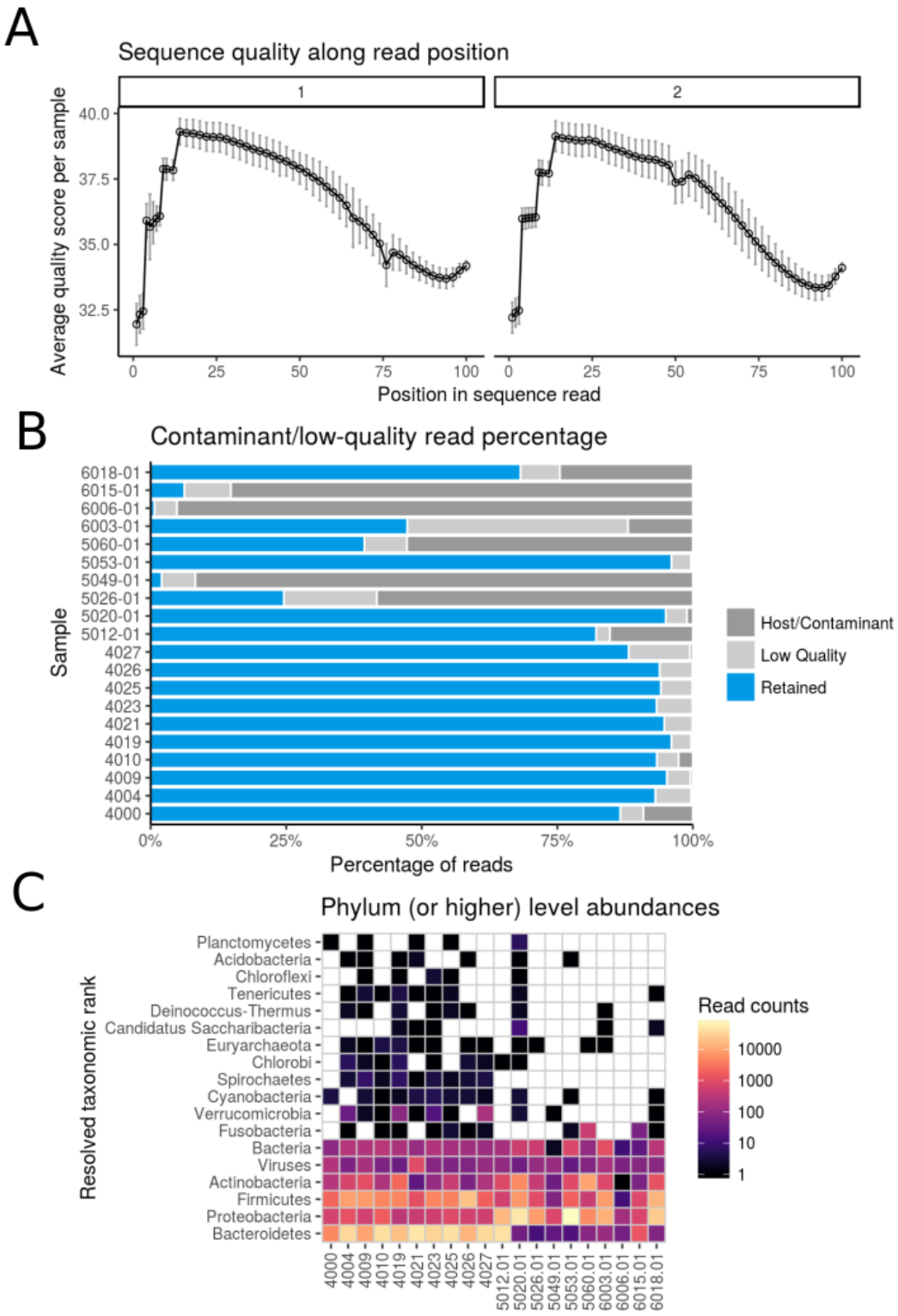
Figure 2



Comparison between the Komplexity software and similar software (BBMask, DUST, and RepeatMasker). The small bar plot in the lower left shows the total nucleotides masked by each tool. The central bar plot shows the number of unique nucleotides masked by every tool combination; each combination is shown by the connected dots below. Bars displaying nucleotides masked by tool combinations that include Komplexity are colored red.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Figure 3



bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Output of example extension sbx_report. (A) shows average read quality per position across all reads. (B) shows the percentage of reads per sample removed during different quality control steps. (C) shows a heatmap of abundances at the phylum level or above from KRAKEN classification.

Tables and Legends

Table 1

Tool	Microsatellite nucleotides masked (%)	CDS nucleotides masked (%)	Speed (kilobase/sec)	Peak memory usage (megabytes)
Komplexity	54.6	0.68	**11,500±1,510**	**4.1±1.6**
RepeatMasker	**57**	0.75	0.65±0.03	607±8.4
BBMask	43	**0.029**	456±79.3	450±11.7
DUST	44.9	0.74	802±12.5	17.3±0.14

Memory usage, speed, and percentage of coding-sequence nucleotides masked for each program. The columns show the percentage of nucleotides (microsatellite or CDS) from reads masked by each tool, as well as the normalized time taken and peak memory usage of each tool while processing the dataset (1.1 megabases). The top-performing tool in each category is shown in bold.

Table 2:

	Sunbeam	SURPI	KneadData	EDGE	ATLAS

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

Quality control	**Adapter trimming**	X (trimmomatic, cutadapt)	X (cutadapt)	X (trimmomatic)	X (FaQCs)	X
	Error correction					X (tadpole)
	Read quality	X (fastqc)	X (cutadapt)	X (fastqc)	X (FaQCs)	X (BBDuk2)
	Host	X (any)	X (human)	X (any)	X (preset list)	X (any)
	Low-complexity	X (Komplexity)	X (DUST)	X (TRF)	X (mono or dinucleotides)	X (BBDuk2)
Sequence analysis	**Reference alignment**	X (extension, bwa)			X (bowtie2, MUMmer+ JBrowse)	X (BBMap)
	Classification	X (KRAKEN)	X (SNAP)		X (GOTTCHA, KRAKEN, MetaPhlAn)	X (DIAMOND)
	Assembly	X (MEGAHit)	X (Minimo)		X (IDBA-UD, SPAdes)	X (MEGAHit, SPAdes)
	ORFs (aa)	X (Prodigal, BLASTp)				X (Prokka)
	Full contig (nt)	X (circularity, BLASTn)	X (RAPSearch)		X (BWA mem)	X (DIAMOND)
	Functional annotation					X
	Phylogeny				X (PhaMe, FastTree/RAxML)	
	Primer design				X (bwa, Primer3)	
Other	**Extension framework**	X (Snakemake)				
	Handles own dependencies	X (conda)	X (bash)	partially (pip)		X (conda)

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

	Modularity	X (Snakemake)			X (Perl modules)	X (Snakemake)
	Results reporting	X (coverage maps, tables)	X (coverage maps, tables)			X (coverage maps, tables)
	Clinical certification		X (CLIA)			

{ Feature comparison for metagenomic pipelines. Tools used by each pipeline: trimmomatic [1]; cutadapt [2]; tadpole [28]; fastqc [29]; FaQCs [30]; BBDuk [31]; DUST [16]; TRF [32]; bwa [3]; bowtie2 [33]; BBMap [34]; KRAKEN [4]; SNAP [35]; MUMmer [36]; JBrowse [37]; GOTTCHA [38]; MetaPhlAn [10]; DIAMOND [39]; FastTree [40]; MEGAHit [5]; SPAdes [41]; Minimo [42]; Prodigal [6]; BLASTp [43]; Prokka [44]; BLASTn [7]; Primer3 [45]; RAPSearch [46]; RAxML [47]; PhaME [48]; conda [49]; Snakemake [50]; samtools [8].

References

1. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics. 2014 Aug 1;30(15):2114-20. doi: 10.1093/bioinformatics/btu170. Epub 2014 Apr 1.
2. Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet. 2011 May 17; doi: 10.14806/ej.17.1.200.
3. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics. 2009 Jul 15;25(14):1754-60. doi: 10.1093/bioinformatics/btp324. Epub 2009 May 18.
4. Wood DE, Salzberg SL. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol. 2014 Mar 3;15(3):R46. doi: 10.1186/gb-2014-15-3-r46.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

5. Li D, Luo R, Liu CM, Leung CM, Ting HF, Sadakane K, Yamashita H, Lam TW. MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices. Methods. 2016 Jun 1;102:3-11. doi: 10.1016/j.ymeth.2016.02.020. Epub 2016 Mar 21.

6. Hyatt D, Chen GL, Locascio PF, Land ML, Larimer FW, Hauser LJ. Prodigal: prokaryotic gene recognition and translation initiation site identification. BMC Bioinformatics. 2010 Mar 8;11:119. doi: 10.1186/1471-2105-11-119.

7. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990 Oct 5;215(3):403-10.

8. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R; Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009 Aug 15;25(16):2078-9. doi: 10.1093/bioinformatics/btp352. Epub 2009 Jun 8.

9. Menzel P, Ng KL, Krogh A. Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nat Commun. 2016 Apr 13;7:11257. doi: 10.1038/ncomms11257.

10. Segata N, Waldron L, Ballarini A, Narasimhan V, Jousson O, Huttenhower C. Metagenomic microbial community profiling using unique clade-specific marker genes. Nat Methods. 2012 Jun 10;9(8):811-4. doi: 10.1038/nmeth.2066.

11. Robinson JT, Thorvaldsdóttir H, Winckler W, Guttman M, Lander ES, Getz G, Mesirov JP. Integrative genomics viewer. Nat Biotechnol. 2011 Jan;29(1):24-6. doi: 10.1038/nbt.1754.

12. Eren AM, Esen ÖC, Quince C, Vineis JH, Morrison HG, Sogin ML, Delmont TO. Anvi'o: an advanced analysis and visualization platform for 'omics data. PeerJ. 2015 Oct 8;3:e1319. doi: 10.7717/peerj.1319. eCollection 2015.

13. Smit AFA, Hubley R & Green P. *RepeatMasker Open-4.0*. 2013-2015 http://www.repeatmasker.org.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

14. Jurka J, Kapitonov VV, Pavlicek A, Klonowski P, Kohany O, Walichiewicz J. Repbase Update, a database of eukaryotic repetitive elements. Cytogenet Genome Res. 2005;110(1-4):462-7.

15. Hubley R, Finn RD, Clements J, Eddy SR, Jones TA, Bao W, Smit AF, Wheeler TJ. The Dfam database of repetitive DNA families. Nucleic Acids Res. 2016 Jan 4;44(D1):D81-9. doi: 10.1093/nar/gkv1272. Epub 2015 Nov 26.

16. Morgulis A, Gertz EM, Schäffer AA, Agarwala R. A fast and symmetric DUST implementation to mask low-complexity DNA sequences. J Comput Biol. 2006 Jun;13(5):1028-40.

17. Joint Genome Institute. BBMask. https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbmask-guide/. Accessed 10 May 2018.

18. Hu X, Yuan J, Shi Y, Lu J, Liu B, Li Z, Chen Y, Mu D, Zhang H, Li N, Yue Z, Bai F, Li H, Fan W. pIRS: Profile-based Illumina pair-end reads simulator. Bioinformatics. 2012 Jun 1;28(11):1533-5. doi: 10.1093/bioinformatics/bts187. Epub 2012 Apr 15.

19. Pruitt KD, Harrow J, Harte RA, Wallin C, Diekhans M, Maglott DR, Searle S, Farrell CM, Loveland JE, Ruef BJ, Hart E, Suner MM, Landrum MJ, Aken B, Ayling S, Baertsch R, Fernandez-Banet J, Cherry JL, Curwen V, Dicuccio M, Kellis M, Lee J, Lin MF, Schuster M, Shkeda A, Amid C, Brown G, Dukhanina O, Frankish A, Hart J, Maidak BL, Mudge J, Murphy MR, Murphy T, Rajan J, Rajput B, Riddick LD, Snow C, Steward C, Webb D, Weber JA, Wilming L, Wu W, Birney E, Haussler D, Hubbard T, Ostell J, Durbin R, Lipman D. The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes. Genome Res. 2009 Jul;19(7):1316-23. doi: 10.1101/gr.080531.108. Epub 2009 Jun 4.

20. NCBI Resource Coordinators. Database Resources of the National Center for Biotechnology Information. Nucleic Acids Res. 2017 Jan 4;45(D1):D12-D17. doi: 10.1093/nar/gkw1071. Epub 2016 Nov 28.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

21. Conway JR, Lex A, Gehlenborg N. UpSetR: an R package for the visualization of intersecting sets and their properties. Bioinformatics. 2017 Sep 15;33(18):2938-2940. doi: 10.1093/bioinformatics/btx364.

22. Naccache SN, Federman S, Veeraraghavan N, Zaharia M, Lee D, Samayoa E, Bouquet J, Greninger AL, Luk KC, Enge B, Wadford DA, Messenger SL, Genrich GL, Pellegrino K, Grard G, Leroy E, Schneider BS, Fair JN, Martínez MA, Isa P, Crump JA, DeRisi JL, Sittler T, Hackett J Jr, Miller S, Chiu CY. A cloud-compatible bioinformatics pipeline for ultrarapid pathogen identification from next-generation sequencing of clinical samples. Genome Res. 2014 Jul;24(7):1180-92. doi: 10.1101/gr.171934.113. Epub 2014 Jun 4.

23. Huttenhower Lab. KneadData. http://huttenhower.sph.harvard.edu/kneaddata. Accessed Apr 20, 2018.

24. Li PE, Lo CC, Anderson JJ,, Davenport KW, Bishop-Lilly KA,, Xu Y, Ahmed S, Feng S, Mokashi VP, Chain PS. Enabling the democratization of the genomics revolution with a fully integrated web-based bioinformatics platform. Nucleic Acids Res. 2017 Jan 9;45(1):67-80. doi: 10.1093/nar/gkw1027. Epub 2016 Nov 28.

25. Richard Allen White III, Joseph Brown, Sean Colby, Christopher C Overall, Joon-Yong Lee, Jeremy Zucker, Kurt R Glaesemann, Christer Jansson, Janet K Jansson ATLAS (Automatic Tool for Local Assembly Structures) - a comprehensive infrastructure for assembly, annotation, and genomic binning of metagenomic and metatranscriptomic data PeerJ Preprints. 10.7287/peerj.preprints.2843v1

26. Florian P Breitwieser, Steven L Salzberg. Pavian: Interactive analysis of metagenomics data for microbiomics and pathogen identification. bioRxiv. doi: /10.1101/084715.

27. Lewis JD, Chen EZ, Baldassano RN, Otley AR, Griffiths AM, Lee D, Bittinger K, Bailey A, Friedman ES, Hoffmann C, Albenberg L, Sinha R, Compher C, Gilroy E, Nessel L, Grant A, Chehoud C, Li H, Wu GD, Bushman FD. Inflammation, Antibiotics, and Diet as Environmental Stressors of the Gut Microbiome in Pediatric Crohn's Disease. Cell Host Microbe. 2015 Oct 14;18(4):489-500. doi: 10.1016/j.chom.2015.09.008.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

28. Joint Genome Institute. Tadpole. https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/tadpole-guide/. Accessed 15 May 2018.

29. Babraham Bioinformatics. FastQC. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ Accessed 10 May 2018.

30. Lo CC, Chain PS,. Rapid evaluation and quality control of next generation sequencing data with FaQCs. BMC Bioinformatics. 2014 Nov 19;15:366. doi: 10.1186/s12859-014-0366-2.

31. Joint Genome Institute. BBDuk. https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/. Accessed 20 March 2018.

32. Benson G. Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Res. 1999 Jan 15;27(2):573-80.

33. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012 Mar 4;9(4):357-9. doi: 10.1038/nmeth.1923.

34. Joint Genome Institute. BBMap. https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbmap-guide/. Accessed 10 May 2018.

35. Matei Zaharia, William J. Bolosky, Kristal Curtis, Armando Fox, David Patterson, Scott Shenker, Ion Stoica, Richard M. Karp, Taylor Sittler. Faster and More Accurate Sequence Alignment with SNAP. arXiv cs.DS. 2011 Nov 23.

36. Delcher AL, Phillippy A, Carlton J, Salzberg SL. Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res. 2002 Jun 1;30(11):2478-83.

37. Skinner ME, Uzilov AV, Stein LD, Mungall CJ, Holmes IH. JBrowse: a next-generation genome browser. Genome Res. 2009 Sep;19(9):1630-8. doi: 10.1101/gr.094607.109. Epub 2009 Jul 1.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

38. Freitas TA, Li PE, Scholz MB, Chain PS. Accurate read-based metagenome characterization using a hierarchical suite of unique signatures. Nucleic Acids Res. 2015 May 26;43(10):e69. doi: 10.1093/nar/gkv180. Epub 2015 Mar 12.

39. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. Nat Methods. 2015 Jan;12(1):59-60. doi: 10.1038/nmeth.3176. Epub 2014 Nov 17.

40. Price MN, Dehal PS, Arkin AP. FastTree 2--approximately maximum-likelihood trees for large alignments. PLoS One. 2010 Mar 10;5(3):e9490. doi: 10.1371/journal.pone.0009490.

41. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD, Pyshkin AV, Sirotkin AV, Vyahhi N, Tesler G, Alekseyev MA, Pevzner PA. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol. 2012 May;19(5):455-77. doi: 10.1089/cmb.2012.0021. Epub 2012 Apr 16.

42. Treangen TJ, Sommer DD, Angly FE, Koren S, Pop M. Next generation sequence assembly with AMOS. Curr Protoc Bioinformatics. 2011 Mar;Chapter 11:Unit 11.8. doi: 10.1002/0471250953.bi1108s33.

43. States DJ, Gish W. Combined use of sequence similarity and codon bias for coding region identification. J Comput Biol. 1994 Spring;1(1):39-50.

44. Seemann T. Prokka: rapid prokaryotic genome annotation. Bioinformatics. 2014 Jul 15;30(14):2068-9. doi: 10.1093/bioinformatics/btu153. Epub 2014 Mar 18.

45. Rozen S, Skaletsky H. Primer3 on the WWW for general users and for biologist programmers. Methods Mol Biol. 2000;132:365-86.

46. Ye Y, Choi JH, Tang H. RAPSearch: a fast protein similarity search tool for short reads. BMC Bioinformatics. 2011 May 15;12:159. doi: 10.1186/1471-2105-12-159.

bioRxiv preprint doi: https://doi.org/10.1101/326363; this version posted May 18, 2018. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC 4.0 International license.

47. Stamatakis A, Ludwig T, Meier H. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics. 2005 Feb 15;21(4):456-63. Epub 2004 Dec 17.

48. Sanaa Afroz Ahmed, Chien-Chi Lo, Po-E Li, Karen W Davenport, Patrick S.G. Chain. From raw reads to trees: Whole genome SNP phylogenetics across the tree of life. bioRxiv. 2015 Nov 19. doi: 10.1101/032250

49. Continuum Analytics, Inc. (dba Anaconda, Inc.). Anaconda. 2017. https://www.anaconda.com.

50. Köster J, Rahmann S. Snakemake--a scalable bioinformatics workflow engine. Bioinformatics. 2012 Oct 1;28(19):2520-2. Epub 2012 Aug 20.