Sunbeam: an extensible pipeline for analyzing metagenomic sequencing experiments

Erik L. Clarke (ecl@pennmedicine.upenn.edu)[1*], Louis J. Taylor (louist@pennmedicine.upenn.edu)[1*], Chunyu Zhao (zhaoc1@email.chop.edu)[2*], Andrew Connell (ancon@pennmedicine.upenn.edu)[1], Frederic D. Bushman (bushman@pennmedicine.upenn.edu)[1], Kyle Bittinger (bittingerk@email.chop.edu)[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.

[+] Corresponding author (bittingerk@email.chop.edu)

Abstract

Background: Analysis of mixed microbial communities using metagenomic sequencing experiments requires multiple preprocessing and analytical steps to interpret the microbial and genetic composition of samples. Analytical steps include quality control, adapter trimming, host decontamination, metagenomic classification, read assembly, and alignment to reference genomes.

Results: We present a modular and user-extensible pipeline called Sunbeam that performs these steps in a consistent and reproducible fashion. It can be installed in a single step, does not require administrative access to the host computer system, and can work with most cluster computing frameworks. We also introduce Komplexity, a software tool to eliminate potentially problematic, low-complexity nucleotide sequences from metagenomic data. A 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 workflow. The pipeline and its extension framework are well documented, in routine use, and regularly updated.

Conclusions: Sunbeam provides a foundation to build more in-depth analyses and to enable comparisons in metagenomic sequencing experiments by removing problematic low complexity reads and standardizing post-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.

Keywords

Sunbeam; shotgun metagenomic sequencing; software; pipeline; quality control

Background

Metagenomic shotgun sequencing involves isolating DNA from a mixed microbial community of interest, then sequencing deeply into DNAs drawn randomly from the mixture. This is in contrast to marker gene sequencing (e.g., the 16S rRNA gene of bacteria), where a specific target gene region is amplified and sequenced. Metagenomic sequencing has enabled critical insights in microbiology [1-9], especially in the study of virus and bacteriophage communities [10-15], and is beginning to be used in clinical diagnosis [16-19]. However, an ongoing challenge is analyzing and interpreting the resulting large datasets in a standard and reliable fashion [20-27].

A common practice to investigate microbial metagenomes is to use Illumina sequencing technology to obtain a large number of short (100-250 base pair) reads from fragmented DNA isolated from a sample of interest. After sequence acquisition, several post-processing steps must be carried out before the sequences can be used to gain insight into the underlying biology [25, 28]. Some steps are common to many sequencing experiments, like quality control and sequencing adapter trimming, while others are unique to shotgun metagenomic sequencing, such as attributing reads to gene ontologies.

Researchers have many tools at their disposal for accomplishing each post-processing step and will frequently encounter multiple parameters in each tool that can 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. Conversely, employing a consistent workflow across studies ensures that experiments are comparable and that the downstream analysis is reproducible, as emphasized in ref [25]. Documentation of software, databases and parameters used is an essential element of this practice; otherwise, the benefits of consistent and reproducible workflows are lost to history.

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 allow simple configuration of software parameters and reference databases; it should provide robust error handling and the ability to resume after interruptions; it should be modular so that unnecessary steps can be skipped or ignored, and it should allow new procedures to be added by the user. The ability to deploy the workflow on a wide range of computer systems ensures that all processing steps can be repeated in different labs with different computing setups and provides flexibility for researchers to choose between computing resources at the institution or in the cloud. Similarly, the ability to record running parameters through the use of configuration files allows for the use of experiment-specific software parameters and serves as documentation for future reference.

Several features contribute to efficient data analysis. It is beneficial if errors or interruptions in the workflow do not require restarting from the beginning—sequencing experiments produce large amounts of data making repeating steps in data processing time-consuming and potentially expensive. In addition, not all steps in a workflow will be necessary for all experiments, and some experiments may require custom processing. To handle experiments appropriately, the workflow should provide an easy way to skip unnecessary steps but run them later if necessary. To make the framework widely useful, users must be able to straightforwardly add new steps into the workflow as needed and share them with others. Several pipelines have been developed that achieve many of these goals [17, 29-31], but did not meet our needs for greater flexibility in processing metagenomic datasets and long-term reproducibility of analyses.

Here, we introduce Sunbeam, an easily-deployable and configurable pipeline that produces a consistent set of post-processed files from metagenomic 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 parallel computing capabilities resulting from its implementation in the Snakemake workflow language [32]. 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 steps to be added at any point in the workflow.

In addition, Sunbeam features custom software that allows it to robustly process data from challenging sample types, including samples with abundant low-quality or host-derived sequences. 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 problematic low-complexity reads before downstream analysis. Reads with low sequence complexity are common in vertebrate-derived samples with low microbial biomass. Microsatellite DNA sequences make up a significant proportion of the human genome and are highly variable between individuals [33-35], compounding the difficulty of removing them by alignment against a single reference genome. We developed Komplexity because existing tools for analyzing nucleotide sequence complexity [36-38] did not meet our needs in terms of speed, removal of spurious hits, and natively processing fastq files.

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 manages and installs all of its own software 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. Software dependencies are automatically installed in an isolated environment to avoid conflicts with existing software outside the pipeline.

Sunbeam architecture

Sunbeam is comprised of a set of discrete steps that take specific files as inputs and produce other files as outputs. Because Sunbeam is implemented using the Snakemake workflow framework, the dependencies between steps are determined at runtime. This allows steps that do not rely on each other to operate independently on separate processors or compute nodes. It also enables robust error handling, because steps that fail or are interrupted do not cause independent steps to fail. In addition, interrupted steps can be resumed without starting from scratch as long as the required input files exist.

Sunbeam is structured in such a way that the output files are grouped conceptually in different folders, providing a logical output folder structure. Users can request specific outputs separately or as a group, and the pipeline will run only the steps required to produce the desired files. This allows the user to skip or re-run any part of the pipeline in a modular fashion.

By default, Sunbeam performs the following preliminary 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 [39] and Cutadapt [40] software. Read pairs surviving quality filtering are kept. Read quality is assessed using FastQC [41] and summarized in separate reports.

2. Low-complexity masking: Sequence complexity in each read is assessed using Komplexity, a kmer-based complexity algorithm newly described below. Reads that fall below a user-customizable threshold are removed. Logs of the number of reads removed are written for later inspection.
3. Host read decontamination: Reads are mapped against a user-specified set of host or contaminant sequences using bwa [42]. Reads that map to any of these sequences within certain identity and length thresholds are removed. The numbers of reads removed are logged for later inspection.

After this initial quality-control process, multiple optional downstream steps can be performed independently. In the *classify* step, the decontaminated and quality-controlled reads are classified taxonomically using Kraken [43]. In the *assembly* step, reads from each sample are assembled into contigs using MEGAHit [44]. Contigs above a pre-specified length are annotated for circularity. Open reading frames (ORFs) are extracted using Prodigal [45]. The contigs [and associated ORFs] are then searched against any number of user-specified nucleotide or protein BLAST [46] databases, using both the entire contig and the putative ORFs. The results are summarized into reports for each sample. Finally, in the independent *mapping* step, quality-controlled reads are mapped using bwa to any number of user-specified reference genomes or gene sets, and the resulting BAM files are sorted and indexed using samtools [47].

Standard outputs from Sunbeam include reads from each step of the quality-control process, taxonomic assignments for each read, contigs built from each sample, gene predictions, and alignment files of all reads to any number of reference genomes. Most rules produce logs of their operation for later inspection and summary.

Installation and versioning

We designed Sunbeam to be as simple as possible to install. To this end, installation requires only copying the software repository and running the installation script. The installation script handles all dependencies and creates the Conda environment, including installing Conda if necessary. At no point are administrative rights required on the host computer. The only requirements are internet connectivity, Linux, and the Bash shell.

We have also incorporated a robust upgrade and semantic versioning system into Sunbeam. Specifically, the set of output files and configuration file options are treated as fixed between major versions of the pipeline to maintain compatibility. Any changes that would change the format or structure of the output folder, or would break compatibility with previous configuration files, only occur during a major version increase (i.e. from version 1.0.0 to version 2.0.0). Minor changes, optimizations, or bugfixes that do not alter the output structure or configuration file may increase the minor or patch version number (i.e. from v1.0.0 to v1.1.0). To prevent unexpected errors, the software checks the version of the configuration file before running to ensure compatibility and will stop if it is from a previous major version. To facilitate upgrading between versions of Sunbeam, the same installation script can also install new versions of the pipeline in-place. We provide a utility to upgrade configuration files between major version changes.

To ensure the stability of the output files and expected behavior of the pipeline, we built a robust integration testing procedure into Sunbeam's development workflow. This integration test checks that Sunbeam is installable on a system, produces the expected set of output files, and correctly handles various configurations and inputs. The test is run through a continuous integration system that is triggered upon any commit to the Sunbeam software repository, and only changes that pass the integration tests are merged into the 'stable' branch used by end-users.

Extensions

The Sunbeam pipeline can be extended by users. Extensions take the form of supplementary *rules* written in the Snakemake format and define the additional steps to be executed. Optionally, two other files may be provided: one listing additional software requirements, and another giving additional configuration options. Extensions can optionally run in a separate software environment, which enables the use of tools with requirements that conflict with Sunbeam's. To integrate these extensions, the user copies the files into Sunbeam's extensions directory, where they are automatically integrated into the workflow during runtime. 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 six lines of code. Because they are integrated directly into the main Sunbeam environment, they have access to the same environmental variables and resources as the primary pipeline, and gain the same error-handling benefits. To make it easy for users to create their own extensions, we provide an extension template on our GitHub page (https://github.com/sunbeam-labs/sbx_template) as well as a number of useful prebuilt extensions available at https://github.com/sunbeam-labs. We created extensions that allow users to run alternate metagenomic read classifiers like Kaiju [48] or MetaPhlAn2 [49], visualize read mappings to reference genomes with IGV [50], and even format Sunbeam outputs for use with downstream analysis pipelines like Anvi'o [51].

Komplexity

We regularly encounter low-complexity sequences comprised of short nucleotide repeats that pose problems for downstream taxonomic assignment and assembly [12, 52], for example by generating spurious alignments to unrelated repeated sequences in database genomes. To avoid these potential artifacts, we created a novel, fast read filter called Komplexity. Komplexity is a stand-alone program implemented in the Rust programming language that is designed to mask or remove 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 provided. Komplexity accepts FASTA and FASTQ files as input and outputs either complexity scores or masked sequences in the input format. As integrated in the Sunbeam workflow, Komplexity assesses the total read complexity and removes reads that fall below the default threshold. Komplexity is also available as a separate open-source program at https://github.com/eclarke/komplexity.

Results and Discussion

Sunbeam implements a core set of commonly-required tasks supplemented by user-built extensions. Even so, the capabilities of Sunbeam compare favorably with existing pipelines such as SURPI (Sequence-based Ultra-Rapid Pathogen Identification) [17], EDGE (Empowering the Development of Genomics Expertise) [29], ATLAS (Automatic Tool for Local Assembly Structures) [30], and KneadData [31], (Table S1). Where Sunbeam's primary advancements lie are in its ease of deployment, extension framework, and novel algorithmic solutions to the issues of low-complexity or host-derived sequence filtering.

To demonstrate the use of Sunbeam on real-world data, we used it on a subset of a previously published dataset of healthy humans or individuals with Crohn's disease [53]. We replaced potentially identifiable human reads with pIRS-simulated human genomic reads [54]. This dataset is available for download at https://zenodo.org/record/1287807. The output of Sunbeam primarily consists of sequence and text files for downstream use, so to demonstrate the potential of the extension system, we created a "report" extension that collects the outputs and visualizes them in a series of plots. These plots, shown in Figure 2, describe metrics such as the average sequence quality at each point in the read (Figure 2A), the total number of contaminant and low-complexity sequences removed for each sample (Figure 2B), and a high-level overview of the taxa found in each sample (Figure 2C). The report extension (https://github.com/sunbeam-labs/sbx_report) demonstrates the ease of building downstream analysis steps into Sunbeam: the report is generated from an R Markdown document. Excluding the R document, the extension is only 12 lines of code long.

Sunbeam's extension framework promotes reproducible analyses and greatly simplifies performing the same type of analysis on multiple datasets. Extension templates, as well as a number of pre-built extensions for metagenomic analysis and visualization software like Anvi'o [51], MetaPhlAn [49], and Pavian [55], are available on our GitHub page (https://github.com/sunbeam-labs).

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 currently exist for filtering low-complexity nucleotide sequences. The gold standard, RepeatMasker [36], uses multiple approaches to identify and mask repeat or low complexity DNA sequences, including querying a database of repetitive DNA elements (either Repbase [56] or Dfam [57]). DUST [37] employs an algorithm which scores and masks nucleotide sequence windows that exceed a particular complexity score threshold (with lower-complexity sequences assigned higher scores) such that no subsequence within the masked region has a 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 [38].

Many of these tools were not optimal for our use with shotgun metagenomic 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 separately. It accepts FASTA/Q files as input, can mask or remove reads below a 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 [54] to simulate Illumina reads from the human conserved coding sequence dataset [58] as well as human microsatellite records from the NCBI nucleotide database [59] 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 (both available at https://zenodo.org/record/1287807). We processed the reads using Komplexity, RepeatMasker, DUST and BBMask and used GNU Time [60] to measure peak memory usage and execution time for six replicates (Table 1). Komplexity and RepeatMasker mask a similar proportion of microsatellite nucleotides, while none of the four tools masks a large proportion of coding nucleotides. 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 (data not shown).

Table 1 - Memory usage, speed, and nucleotides masked for each program.

Tool	**Microsatellite nucleotides masked (%)**	**Conserved coding sequence 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

Columns show the percentage of nucleotides (microsatellite or conserved coding sequence) 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.

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 [61] (Figure 3). 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. Komplexity in combination with

Sunbeam's standard host removal system resulted in the removal of over 99% of the total simulated microsatellite reads.

Conclusions

Here we introduce Sunbeam, a Snakemake-based pipeline for analyzing shotgun metagenomic data with a focus on reproducible analysis, ease of deployment and use. We compare Sunbeam with other pipelines for metagenomic analysis and note several favorable features. 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 and use simplify the processing of shotgun metagenomic sequence data, while its extension framework and thorough quality control enable robust and reproducible analyses. We have already used Sunbeam in multiple published [23, 52, 62, 86] and ongoing studies. As a case study for Sunbeam's ease-of-use and robust deployability, we featured Sunbeam at a metagenomics workshop at the University of Pennsylvania in the summer of 2017. All participants successfully installed and ran the full pipeline on sample datasets, which emphasizes the ease of deploying and using Sunbeam.

Availability and requirements

Project name: Sunbeam

Project home page: https://github.com/sunbeam-labs/sunbeam

Operating system: Linux

Programming languages: Python, Rust and Snakemake

License: GPLv3

Restrictions to use by non-academics: No

List of abbreviations

ATLAS: Automatic Tool for Local Assembly Structures; BAM: binary alignment map; BLAST: Basic Local Alignment Search Tool; EDGE: Empowering the Development of Genomics Expertise; GPL: (GNU) General Public License; ORF(s): open reading frame(s); SDUST: Symmetric DUST; SURPI: Sequence-based Ultra-Rapid Pathogen Identification

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and material

Sunbeam is available at https://github.com/sunbeam-labs/sunbeam. Komplexity is available at https://github.com/eclarke/komplexity. Pre-built extensions referenced can be found at https://github.com/sunbeam-labs/. The dataset of simulated microsatellite and conserved coding sequence reads, as well as the example dataset analyzed in Figure 2, are archived in Zenodo at https://zenodo.org/record/1287807.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the NIH grants U01HL112712 (Site-Specific Genomic Research in Alpha-1 Antitrypsin Deficiency and Sarcoidosis (GRADS) Study), R01HL113252, and R61HL137063, and received assistance from the Penn Center for AIDS Research (P30AI045008), T32 Training Grant (T32AI007324, LJT), and the PennCHOP Microbiome Program (Tobacco Formula grant under the Commonwealth Universal Research Enhancement (C.U.R.E) program with the grant number SAP # 4100068710).

Authors' contributions

ELC, CZ, FDB, and KB conceived and designed Sunbeam. ELC, CZ, AC, and KB developed Sunbeam. ELC and LJT conceived and developed Komplexity. LJT performed the low-complexity sequence masking analysis. ELC, LJT, and KB wrote the manuscript. All authors read, improved, and approved the final manuscript.

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.

Figure Legends

Figure 1

Flowchart of inputs, processes and outputs for standard steps in the Sunbeam metagenomics pipeline.

Figure 2

Output of example extension sbx_report, which produces plots of (A) average read quality per position across all reads, (B) percentage of reads per sample removed during quality control steps, and (C) relative abundances at the phylum level or above from KRAKEN classification.

Figure 3

Comparison between Komplexity 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.

Supplementary table legends

Table S1 (Additional File 1):

Feature comparison for metagenomic pipelines. Tools used by each pipeline: trimmomatic [39]; cutadapt [40]; tadpole [63]; fastqc [41]; FaQCs [64]; BBDuk2 [65]; DUST [37]; TRF [66]; bwa [42]; bowtie2 [67]; BBMap [68]; KRAKEN [43]; SNAP [69]; MUMmer [70]; JBrowse [71]; GOTTCHA [72]; MetaPhlAn [49]; DIAMOND [73]; FastTree [74]; MEGAHit ; SPAdes [75]; Minimo [76]; Prodigal [45]; BLASTp [46]; Prokka [77]; BLASTn [46]; eggNOG [78]; ENZYME [79]; dbCAN [80]; Primer3 [81]; RAPSearch [82]; RAxML [83]; PhaME [84]; conda [85]; Snakemake [32]; samtools [47].

References

1. Turnbaugh PJ, Ley RE, Hamady M, Fraser-Liggett CM, Knight R, Gordon JI: **The Human Microbiome Project**. *Nature* 2007, **449**(7164):804-810.
2. Muegge BD, Kuczynski J, Knights D, Clemente JC, Gonzalez A, Fontana L, Henrissat B, Knight R, Gordon JI: **Diet Drives Convergence in Gut Microbiome Functions Across Mammalian Phylogeny and Within Humans**. *Science* 2011, **332**(6032):970-974.
3. Turnbaugh PJ, Ley RE, Mahowald MA, Magrini V, Mardis ER, Gordon JI: **An obesity-associated gut microbiome with increased capacity for energy harvest**. *Nature* 2006, **444**(7122):1027-1131.
4. Morgan XC, Tickle TL, Sokol H, Gevers D, Devaney KL, Ward DV, Reyes JA, Shah SA, LeLeiko N, Snapper SB *et al*: **Dysfunction of the intestinal microbiome in inflammatory bowel disease and treatment**. *Genome Biology* 2012, **13**(9).
5. Yatsunenko T, Rey FE, Manary MJ, Trehan I, Dominguez-Bello MG, Contreras M, Magris M, Hidalgo G, Baldassano RN, Anokhin AP *et al*: **Human gut microbiome viewed across age and geography**. *Nature* 2012, **486**(7402):222-227.
6. Eisen JA, Abubucker S, Segata N, Goll J, Schubert AM, Izard J, Cantarel BL, Rodriguez-Mueller B, Zucker J, Thiagarajan M *et al*: **Metabolic Reconstruction for Metagenomic Data and Its Application to the Human Microbiome**. *PLoS Computational Biology* 2012, **8**(6).
7. Fierer N, Leff JW, Adams BJ, Nielsen UN, Bates ST, Lauber CL, Owens S, Gilbert JA, Wall DH, Caporaso JG: **Cross-biome metagenomic analyses of soil microbial communities and their functional attributes**. *Proceedings of the National Academy of Sciences* 2012, **109**(52):21390-21395.
8. Dinsdale EA, Edwards RA, Hall D, Angly F, Breitbart M, Brulc JM, Furlan M, Desnues C, Haynes M, Li L *et al*: **Functional metagenomic profiling of nine biomes**. *Nature* 2008, **452**(7187):629-632.
9. Lee STM, Kahn SA, Delmont TO, Shaiber A, Esen ÖC, Hubert NA, Morrison HG, Antonopoulos DA, Rubin DT, Eren AM: **Tracking microbial colonization in fecal microbiota transplantation experiments via genome-resolved metagenomics**. *Microbiome* 2017, **5**(1).
10. Breitbart M, Hewson I, Felts B, Mahaffy JM, Nulton J, Salamon P, Rohwer F: **Metagenomic Analyses of an Uncultured Viral Community from Human Feces**. *Journal of Bacteriology* 2003, **185**(20):6220-6223.
11. Edwards RA, Rohwer F: **Opinion: Viral metagenomics**. *Nature Reviews Microbiology* 2005, **3**(6):504-510.
12. Abbas AA, Diamond JM, Chehoud C, Chang B, Kotzin JJ, Young JC, Imai I, Haas AR, Cantu E, Lederer DJ *et al*: **The Perioperative Lung Transplant Virome: Torque Teno Viruses Are Elevated in Donor Lungs and Show Divergent Dynamics in Primary Graft Dysfunction**. *Am J Transplant* 2017, **17**(5):1313-1324.
13. Emerson JB, Thomas BC, Andrade K, Allen EE, Heidelberg KB, Banfield JF: **Dynamic Viral Populations in Hypersaline Systems as Revealed by Metagenomic Assembly**. *Applied and Environmental Microbiology* 2012, **78**(17):6309-6320.
14. Ma Y, Madupu R, Karaoz U, Nossa CW, Yang L, Yooseph S, Yachimski PS, Brodie EL, Nelson KE, Pei Z: **Human Papillomavirus Community in Healthy Persons, Defined by Metagenomics Analysis of Human Microbiome Project Shotgun Sequencing Data Sets**. *Journal of Virology* 2014, **88**(9):4786-4797.

15. Minot S, Bryson A, Chehoud C, Wu GD, Lewis JD, Bushman FD: **Rapid evolution of the human gut virome**. *Proc Natl Acad Sci U S A* 2013, **110**(30):12450-12455.
16. Greninger AL, Naccache SN, Federman S, Yu G, Mbala P, Bres V, Stryke D, Bouquet J, Somasekar S, Linnen JM *et al*: **Rapid metagenomic identification of viral pathogens in clinical samples by real-time nanopore sequencing analysis**. *Genome Medicine* 2015, **7**(1).
17. Naccache SN, Federman S, Veeraraghavan N, Zaharia M, Lee D, Samayoa E, Bouquet J, Greninger AL, Luk KC, Enge B *et al*: **A cloud-compatible bioinformatics pipeline for ultrarapid pathogen identification from next-generation sequencing of clinical samples**. *Genome Research* 2014, **24**(7):1180-1192.
18. Virgin Herbert W, Todd John A: **Metagenomics and Personalized Medicine**. *Cell* 2011, **147**(1):44-56.
19. Houldcroft CJ, Beale MA, Breuer J: **Clinical and biological insights from viral genome sequencing**. *Nature Reviews Microbiology* 2017, **15**(3):183-192.
20. Meisel JS, Hannigan GD, Tyldsley AS, SanMiguel AJ, Hodkinson BP, Zheng Q, Grice EA: **Skin Microbiome Surveys Are Strongly Influenced by Experimental Design**. *Journal of Investigative Dermatology* 2016, **136**(5):947-956.
21. Weiss S, Amir A, Hyde ER, Metcalf JL, Song SJ, Knight R: **Tracking down the sources of experimental contamination in microbiome studies**. *Genome Biology* 2014, **15**(12).
22. Kim D, Hofstaedter CE, Zhao C, Mattei L, Tanes C, Clarke E, Lauder A, Sherrill-Mix S, Chehoud C, Kelsen J *et al*: **Optimizing methods and dodging pitfalls in microbiome research**. *Microbiome* 2017, **5**(1).
23. Lauder AP, Roche AM, Sherrill-Mix S, Bailey A, Laughlin AL, Bittinger K, Leite R, Elovitz MA, Parry S, Bushman FD: **Comparison of placenta samples with contamination controls does not provide evidence for a distinct placenta microbiota**. *Microbiome* 2016, **4**(1):29.
24. Nayfach S, Pollard KS: **Toward Accurate and Quantitative Comparative Metagenomics**. *Cell* 2016, **166**(5):1103-1116.
25. Knight R, Vrbanac A, Taylor BC, Aksenov A, Callewaert C, Debelius J, Gonzalez A, Kosciolek T, McCall LI, McDonald D *et al*: **Best practices for analysing microbiomes**. *Nat Rev Microbiol* 2018.
26. Delmont TO, Eren AM: **Identifying contamination with advanced visualization and analysis practices: metagenomic approaches for eukaryotic genome assemblies**. *PeerJ* 2016, **4**:e1839.
27. Kjartansdóttir KR, Friis-Nielsen J, Asplund M, Mollerup S, Mourier T, Jensen RH, Hansen TA, Rey-Iglesia A, Richter SR, Alquezar-Planas DE *et al*: **Traces of ATCV-1 associated with laboratory component contamination**. *Proceedings of the National Academy of Sciences* 2015, **112**(9):E925-E926.
28. Quince C, Walker AW, Simpson JT, Loman NJ, Segata N: **Shotgun metagenomics, from sampling to analysis**. *Nat Biotechnol* 2017, **35**(9):833-844.
29. Li P-E, Lo C-C, Anderson JJ, Davenport KW, Bishop-Lilly KA, Xu Y, Ahmed S, Feng S, Mokashi VP, Chain PSG: **Enabling the democratization of the genomics revolution with a fully integrated web-based bioinformatics platform**. *Nucleic Acids Research* 2017, **45**(1):67-80.
30. White Iii RA, Brown J, Colby S, Overall CC, Lee J-Y, Zucker J, Glaesemann KR, Jansson C, Jansson JK: **ATLAS (Automatic Tool for Local Assembly Structures) - a comprehensive infrastructure for assembly, annotation, and genomic binning of metagenomic and metatranscriptomic data**. *PeerJ Preprints* 2017.
31. **KneadData**. *BioBakery* 2017, **https://bitbucket.org/biobakery/kneaddata**.
32. Koster J, Rahmann S: **Snakemake--a scalable bioinformatics workflow engine**. *Bioinformatics* 2012, **28**(19):2520-2522.

33. Subramanian S, Mishra RK, Singh L: **Genome-wide analysis of microsatellite repeats in humans: their abundance and density in specific genomic regions**. *Genome Biology* 2003, **4**(2).
34. Banchs I, Bosch A, Guimerà J, Lázaro C, Puig A, Estivill X: **New alleles at microsatellite loci in CEPH families mainly arise from somatic mutations in the lymphoblastoid cell lines**. *Human Mutation* 1994, **3**(4):365-372.
35. Payseur BA, Nachman MW: **Microsatellite variation and recombination rate in the human genome**. *Genetics* 2000, **156**(3):1285-1298.
36. Smit A, Hubley R, Green P: **RepeatMasker Open-4.0.** 2013-2015, **http://www.repeatmasker.org**.
37. Morgulis A, Gertz EM, Schaffer AA, Agarwala R: **A fast and symmetric DUST implementation to mask low-complexity DNA sequences**. *J Comput Biol* 2006, **13**(5):1028-1040.
38. JGI: **BBMask**. **https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbmask-guide/**.
39. Bolger AM, Lohse M, Usadel B: **Trimmomatic: a flexible trimmer for Illumina sequence data**. *Bioinformatics* 2014, **30**(15):2114-2120.
40. Martin M: **Cutadapt removes adapter sequences from high-throughput sequencing reads**. *EMBnetjournal* 2011, **17**(1).
41. BabrahamBioinformatics: **FastQC**. **https://www.bioinformatics.babraham.ac.uk/projects/fastqc/**.
42. Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform**. *Bioinformatics* 2009, **25**(14):1754-1760.
43. Wood DE, Salzberg SL: **Kraken: ultrafast metagenomic sequence classification using exact alignments**. *Genome Biology* 2014, **15**(3).
44. Li D, Luo R, Liu C-M, Leung C-M, Ting H-F, Sadakane K, Yamashita H, Lam T-W: **MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices**. *Methods* 2016, **102**:3-11.
45. Hyatt D, Chen G-L, LoCascio PF, Land ML, Larimer FW, Hauser LJ: **Prodigal: prokaryotic gene recognition and translation initiation site identification**. *BMC Bioinformatics* 2010, **11**(1).
46. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool**. *J Mol Biol* 1990, **215**(3):403-410.
47. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R: **The Sequence Alignment/Map format and SAMtools**. *Bioinformatics* 2009, **25**(16):2078-2079.
48. Menzel P, Ng KL, Krogh A: **Fast and sensitive taxonomic classification for metagenomics with Kaiju**. *Nature Communications* 2016, **7**.
49. Truong DT, Franzosa EA, Tickle TL, Scholz M, Weingart G, Pasolli E, Tett A, Huttenhower C, Segata N: **MetaPhlAn2 for enhanced metagenomic taxonomic profiling**. *Nat Methods* 2015, **12**(10):902-903.
50. Robinson JT, Thorvaldsdóttir H, Winckler W, Guttman M, Lander ES, Getz G, Mesirov JP: **Integrative genomics viewer**. *Nature Biotechnology* 2011, **29**(1):24-26.
51. 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, **3**.
52. Clarke EL, Lauder AP, Hofstaedter CE, Hwang Y, Fitzgerald AS, Imai I, Biernat W, Rekawiecki B, Majewska H, Dubaniewicz A *et al*: **Microbial Lineages in Sarcoidosis. A Metagenomic Analysis Tailored for Low-Microbial Content Samples**. *Am J Respir Crit Care Med* 2018, **197**(2):225-234.
53. Lewis James D, Chen Eric Z, Baldassano Robert N, Otley Anthony R, Griffiths Anne M, Lee D, Bittinger K, Bailey A, Friedman Elliot S, Hoffmann C *et al*: **Inflammation, Antibiotics, and Diet as Environmental Stressors of the Gut Microbiome in Pediatric Crohn's Disease**. *Cell Host & Microbe* 2015, **18**(4):489-500.

54. Hu X, Yuan J, Shi Y, Lu J, Liu B, Li Z, Chen Y, Mu D, Zhang H, Li N *et al*: **pIRS: Profile-based Illumina pair-end reads simulator**. *Bioinformatics* 2012, **28**(11):1533-1535.

55. Breitwieser FP, Salzberg SL: **Pavian: Interactive analysis of metagenomics data for microbiomics and pathogen identification**. *bioRxiv* 2016.

56. 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):462-467.

57. Hubley R, Finn RD, Clements J, Eddy SR, Jones TA, Bao W, Smit AFA, Wheeler TJ: **The Dfam database of repetitive DNA families**. *Nucleic Acids Research* 2016, **44**(D1):D81-D89.

58. Pruitt KD, Harrow J, Harte RA, Wallin C, Diekhans M, Maglott DR, Searle S, Farrell CM, Loveland JE, Ruef BJ *et al*: **The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes**. *Genome Research* 2009, **19**(7):1316-1323.

59. Coordinators NR: **Database Resources of the National Center for Biotechnology Information**. *Nucleic Acids Research* 2017, **45**(D1):D12-D17.

60. **GNU Time**. **https://www.gnu.org/software/time/**.

61. Conway JR, Lex A, Gehlenborg N: **UpSetR: an R package for the visualization of intersecting sets and their properties**. *Bioinformatics* 2017, **33**(18):2938-2940.

62. Taylor JM, Clarke EL, Baker K, Lauder A, Kim D, Bailey A, Wu GD, Collman RG, Doyle-Meyers L, Russell-Lodrigue K *et al*: **Evaluation of a therapy for Idiopathic Chronic Enterocolitis in rhesus macaques (Macaca mulatta) and linked microbial community correlates**. *PeerJ* 2018, **6**.

63. JGI: **Tadpole**. **https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/tadpole-guide/**.

64. Lo C-C, Chain PSG: **Rapid evaluation and quality control of next generation sequencing data with FaQCs**. *BMC Bioinformatics* 2014, **15**(1).

65. JGI: **BBDuk**. **https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/**.

66. Benson G: **Tandem repeats finder: a program to analyze DNA sequences**. *Nucleic Acids Res* 1999, **27**(2):573-580.

67. Langmead B, Salzberg SL: **Fast gapped-read alignment with Bowtie 2**. *Nature Methods* 2012, **9**(4):357-359.

68. JGI: **BBMap**. **https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbmap-guide/**.

69. Zaharia M, Bolosky WJ, Curtis K, Fox A, Patterson D, Shenker S, Stoica I, M. Karp R, Sittler T: **Faster and More Accurate Sequence Alignment with SNAP**. *arXiv* 2011.

70. Delcher AL, Phillippy A, Carlton J, Salzberg SL: **Fast algorithms for large-scale genome alignment and comparison**. *Nucleic Acids Res* 2002, **30**(11):2478-2483.

71. Skinner ME, Uzilov AV, Stein LD, Mungall CJ, Holmes IH: **JBrowse: a next-generation genome browser**. *Genome Res* 2009, **19**(9):1630-1638.

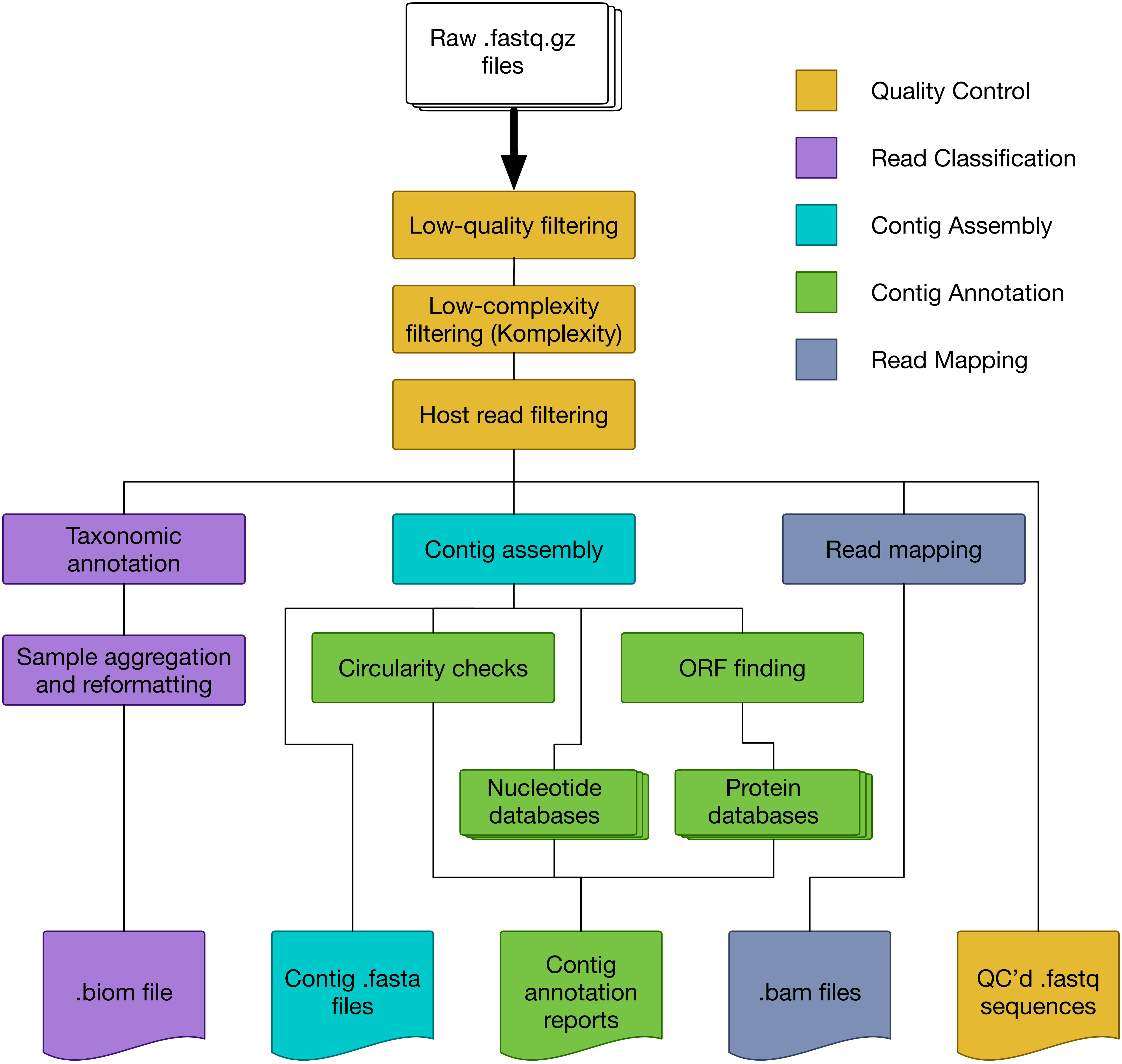
72. Freitas Tracey Allen K, Li P-E, Scholz MB, Chain Patrick SG: **Accurate read-based metagenome characterization using a hierarchical suite of unique signatures**. *Nucleic Acids Research* 2015, **43**(10):e69-e69.

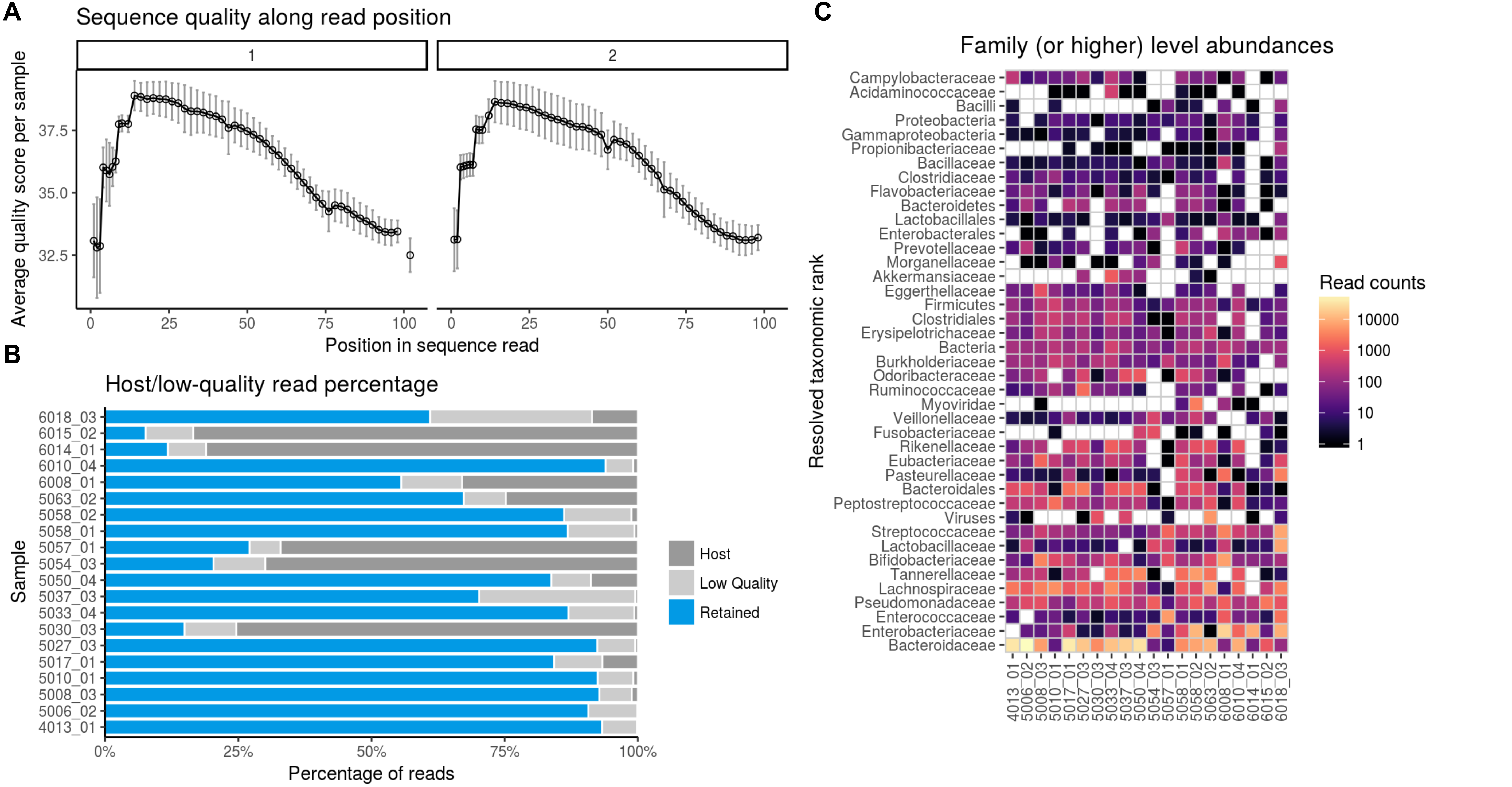
73. Buchfink B, Xie C, Huson DH: **Fast and sensitive protein alignment using DIAMOND**. *Nature Methods* 2014, **12**(1):59-60.

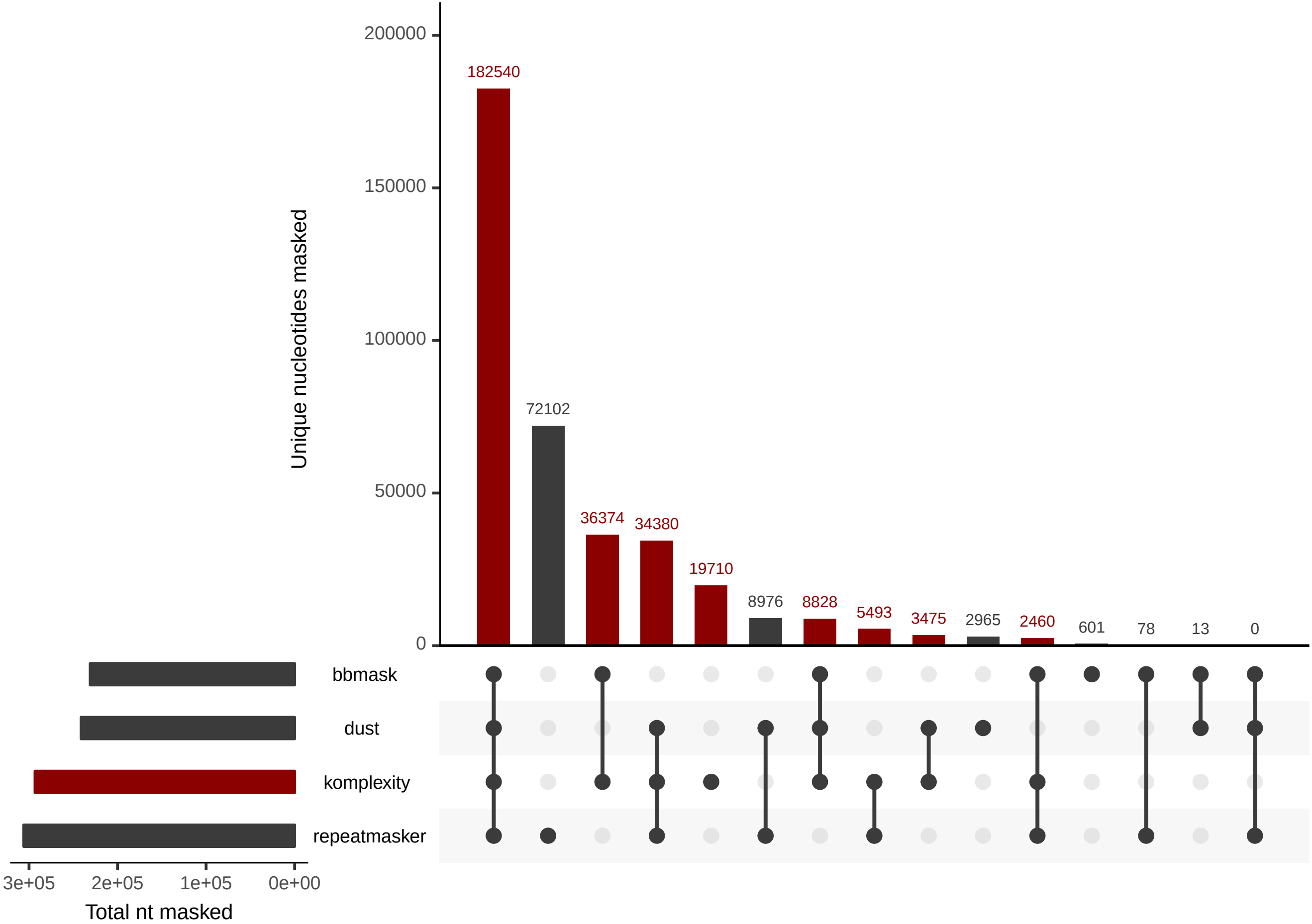
74. Price MN, Dehal PS, Arkin AP: **FastTree 2--approximately maximum-likelihood trees for large alignments**. *PLoS One* 2010, **5**(3):e9490.

75. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD *et al*: **SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing**. *J Comput Biol* 2012, **19**(5):455-477.

76. Treangen TJ, Sommer DD, Angly FE, Koren S, Pop M: **Next generation sequence assembly with AMOS**. *Curr Protoc Bioinformatics* 2011, **Chapter 11**:Unit 11 18.
77. Seemann T: **Prokka: rapid prokaryotic genome annotation**. *Bioinformatics* 2014, **30**(14):2068-2069.
78. Jensen LJ, Julien P, Kuhn M, von Mering C, Muller J, Doerks T, Bork P: **eggNOG: automated construction and annotation of orthologous groups of genes**. *Nucleic Acids Res* 2008, **36**(Database issue):D250-254.
79. Bairoch A: **The ENZYME database in 2000**. *Nucleic Acids Res* 2000, **28**(1):304-305.
80. Yin Y, Mao X, Yang J, Chen X, Mao F, Xu Y: **dbCAN: a web resource for automated carbohydrate-active enzyme annotation**. *Nucleic Acids Res* 2012, **40**(Web Server issue):W445-451.
81. Rozen S, Skaletsky H: **Primer3 on the WWW for general users and for biologist programmers**. *Methods Mol Biol* 2000, **132**:365-386.
82. Ye Y, Choi JH, Tang H: **RAPSearch: a fast protein similarity search tool for short reads**. *BMC Bioinformatics* 2011, **12**:159.
83. Stamatakis A, Ludwig T, Meier H: **RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees**. *Bioinformatics* 2005, **21**(4):456-463.
84. Ahmed SA, Lo C-C, Li P-E, Davenport KW, Chain PSG: **From raw reads to trees: Whole genome SNP phylogenetics across the tree of life**. *bioRxiv* 2015.
85. Anaconda INC: **conda**. **https://anaconda.org/**.
86. Leiby JS, Mccormick K, Sherrill-Mix S, Clarke EL, Kessler LR, Taylor LJ, Hofstaedter CE, Roche AM, Mattei LM, Bittinger K, Elovitz MA, Leite R, Parry S, Bushman FD. **Lack of detection of a human placenta microbiome in samples from preterm and term deliveries.** *Microbiome* 2018. **6**(1).


Raw .fastq.gz files
Quality Control
Read Classification
Contig Assembly
Contig Annotation
Read Mapping
Low-quality filtering
Low-complexity filtering (Komplexity)
Host read filtering
Taxonomic annotation
Contig assembly
Read mapping
Sample aggregation and reformatting
Circularity checks
ORF finding
Nucleotide databases
Protein databases
.biom file
Contig .fasta files
Contig annotation reports
.bam files
QC'd .fastq sequences


A
Sequence quality along read position
1
2
Average quality score per sample
37.5
35.0
32.5
0
25
50
75
100
Position in sequence read
B
Host/low-quality read percentage
6018_03
6015_02
6014_01
6010_04
6008_01
5063_02
5058_02
5058_01
5057_01
5054_03
5050_04
5037_03
5033_04
5030_03
5027_03
5017_01
5010_01
5008_03
5006_02
4013_01
Sample
0%
25%
50%
75%
100%
Percentage of reads
Host
Low Quality
Retained
C
Family (or higher) level abundances
Campylobacteraceae
Acidaminococcaceae
Bacilli
Proteobacteria
Gammaproteobacteria
Propionibacteriaceae
Bacillaceae
Clostridiaceae
Flavobacteriaceae
Bacteroidetes
Lactobacillales
Enterobacterales
Prevotellaceae
Morganellaceae
Akkermansiaceae
Eggerthellaceae
Firmicutes
Clostridiales
Erysipelotrichaceae
Bacteria
Burkholderiaceae
Odoribacteraceae
Ruminococcaceae
Myoviridae
Veillonellaceae
Fusobacteriaceae
Rikenellaceae
Eubacteriaceae
Pasteurellaceae
Bacteroidales
Peptostreptococcaceae
Viruses
Streptococcaceae
Lactobacillaceae
Bifidobacteriaceae
Tannerellaceae
Lachnospiraceae
Pseudomonadaceae
Enterococcaceae
Enterobacteriaceae
Bacteroidaceae
Resolved taxonomic rank
4013_01
5006_02
5008_03
5010_01
5017_01
5027_03
5030_03
5033_04
5037_03
5050_04
5054_03
5057_01
5058_01
5058_02
5063_02
6008_01
6010_04
6014_01
6015_02
6018_03
Read counts
10000
1000
100
10
1


Unique nucleotides masked
200000
150000
100000
50000
0
182540
72102
36374
34380
19710
8976
8828
5493
3475
2965
2460
601
78
13
0
bbmask
dust
komplexity
repeatmasker
3e+05
2e+05
1e+05
0e+00
Total nt masked