nature protocols

PROTOCOL

https://doi.org/10.1038/s41596-022-00738-y

Check for updates

Metagenome analysis using the Kraken software suite

Jennifer Lu[1,2,6]✉, Natalia Rincon[1,2,6], Derrick E. Wood[2,3], Florian P. Breitwieser[2], Christopher Pockrandt[2], Ben Langmead[3], Steven L. Salzberg[1,2,3,4] and Martin Steinegger[5]✉

Metagenomic experiments expose the wide range of microscopic organisms in any microbial environment through high-throughput DNA sequencing. The computational analysis of the sequencing data is critical for the accurate and complete characterization of the microbial community. To facilitate efficient and reproducible metagenomic analysis, we introduce a step-by-step protocol for the Kraken suite, an end-to-end pipeline for the classification, quantification and visualization of metagenomic datasets. Our protocol describes the execution of the Kraken programs, via a sequence of easy-to-use scripts, in two scenarios: (1) quantification of the species in a given metagenomics sample; and (2) detection of a pathogenic agent from a clinical sample taken from a human patient. The protocol, which is executed within 1–2 h, is targeted to biologists and clinicians working in microbiome or metagenomics analysis who are familiar with the Unix command-line environment.

Introduction

Metagenomics sequencing has greatly improved our understanding of the microscopic world by revealing a vast range of microbial organisms that were previously unobserved, many of which cannot be grown in laboratory cultures[1]. Metagenomics experiments take many forms, two of which can be broadly categorized as either microbiome experiments or pathogen identification experiments. In microbiome experiments, researchers evaluate all the microbial organisms identified in a given sample, often with the goal of describing what is present. In contrast, in a pathogen identification experiment, researchers focus on identifying one or a few pathogenic microbes, with the goal of diagnosing an infection. The suite of tools in the Kraken package were developed to cover many of the bioinformatics needs of both microbiome and pathogen metagenomics experiments. These tools include Kraken[2], KrakenUniq[3], Kraken 2 (ref. [4]), Kraken2Uniq (based on KrakenUniq and included in the Kraken 2 codebase), Bracken[5], KrakenTools and Pavian[6].

Microbiome experiments begin with (1) removing host DNA and then (2) classifying a set of sequencing reads, with each read assigned to a taxonomic category (species, genus or higher-level taxa), followed by (3) computing the relative abundance of different species in the sample. When computing relative abundance, researchers sometimes focus on a limited number of 'marker' genes, classifying only the reads that align within these genes. The most widely used marker gene is ribosomal RNA, but protein-coding genes can also be used, particularly those that are expected to be present in exactly one copy per cell. However, many studies prefer to analyze all the sequencing reads collected from a sample, which is the strategy we focus on in this protocol. Following characterization of the original data, downstream analyses may include (4) statistical methods for comparing the microbial compositions of different environments or (5) visualization methods for understanding microbial compositions. Kraken[2] and Kraken 2 (ref. [4]) (an improved version of Kraken) were previously developed by some of us for rapid, accurate classification of sequencing reads. Bracken[5] was developed to work in conjunction with Kraken to compute species abundance using Kraken classification results. Finally, KrakenTools (https://github.com/jenniferlu717/KrakenTools) and Pavian[6] provide a comprehensive set of tools for downstream statistical analysis and visualization of the classification and abundance estimation results.

[1]Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, USA. [2]Center for Computational Biology, Whiting School of Engineering, Johns Hopkins University, Baltimore, MD, USA. [3]Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA. [4]Department of Biostatistics, Johns Hopkins University, Baltimore, MD, USA. [5]School of Biological Sciences and Institute of Molecular Biology & Genetics, Seoul National University, Seoul, Republic of Korea. [6]These authors contributed equally: Jennifer Lu, Natalia Rincon.
✉e-mail: jennifer.lu717@gmail.com; martin.steinegger@snu.ac.kr

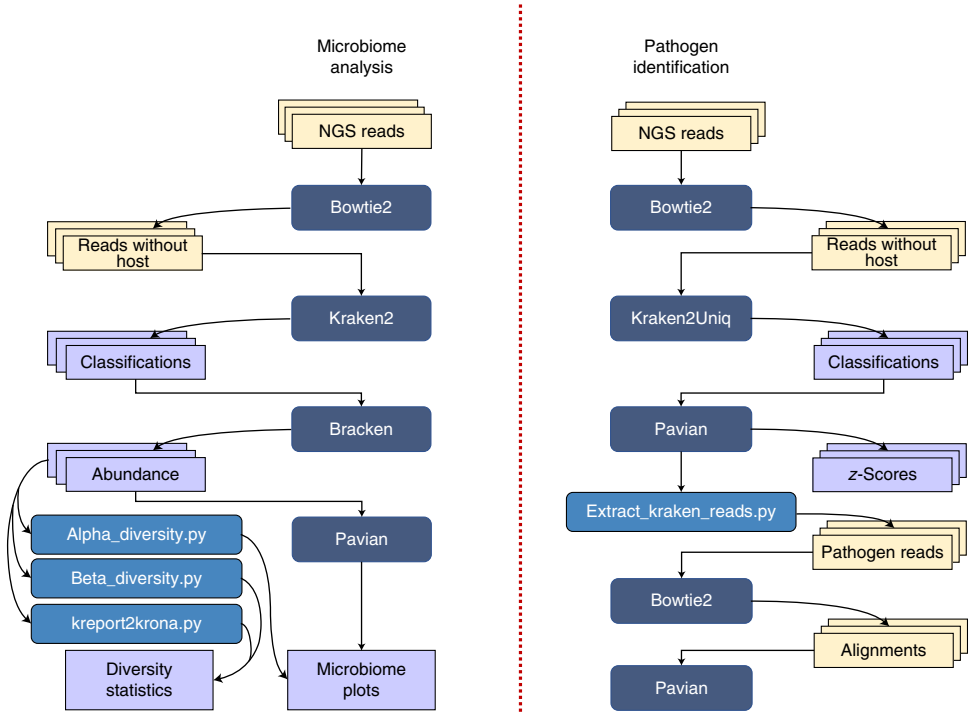

Fig. 1 | Protocol workflow. Overview of the (1) microbiome analysis and (2) pathogen identification workflows. In Procedure 1, we try to estimate the abundance of species in microbiome samples and compute the diversity changes between them. We start with multiple sets of reads from a microbiome before and after fecal transfer. All samples are classified using Kraken 2. Bracken takes the classified read counts and estimates the abundance of each taxon in the sample. Pavian can be used to explore and visualize this sample to spot the difference. Additionally, `alpha_diversity.py` can be used to quantify the diversity in a sample, and `beta_diversity.py` can be used to compare diversity across samples. In Procedure 2, we try to detect an infectious agent using Illumina short reads. We start with a sample from the infection site and (ideally) a negative control. As a first step, host DNA is removed by excluding all reads aligning to the host genome using Bowtie 2. This step usually removes a large fraction of the reads. Remaining reads are then classified by Kraken2Uniq against a reference database, and the taxonomic reports are compared using Pavian. Pavian can distinguish large read count differences between controls and infected samples using *z*-statistics. For all potential pathogen candidates, reads can be extracted using `extract_kraken_reads.py`. NGS, next-generation sequencing.

In pathogen identification experiments, researchers and clinicians are interested in identifying pathogenic microbes that might be the cause of a harmful infection. Pathogen identification studies conceptually include the following steps: (1) removal of host DNA from the microbial reads; (2) classification of the remaining microbial reads; (3) comparison of sample reads against control samples; and (4) validation of pathogen classifications. The second and third steps are essentially the same as those done for microbiome experiments, while the first and last steps are needed to ensure the accuracy of any pathogen that is identified. For removal of host DNA, one can use Bowtie 2 (ref. [7]) (also developed by some of the authors of this protocol) as a fast, sensitive aligner that can compare sequencing reads with the human reference genome. Classification of the remaining reads is then accomplished by using KrakenUniq[3] and Kraken2Uniq. Finally, KrakenTools and Pavian[6] cover the remaining steps of pathogen identification experiments, assisting in the identification and verification of potential pathogens.

Here we describe in detail how to use the Kraken suite to analyze metagenomic data for both microbiome and pathogen identification experiments. Figure 1 outlines the procedures for each of the two types of study.

Overview of the protocol

The procedures described here focus on two major categories of metagenomic experiments: microbiome analysis (Procedure 1) and pathogen identification (Procedure 2). We detail the protocol steps using DNA samples from the human gut and cornea. However, these procedures may also be applied

to RNA data as well as other biomes and species, without any additional modifications. For both procedures, human reads were already removed before analysis. However, we include the host removal step in the protocol to demonstrate how to remove host reads using Bowtie 2 (ref. [7]).

For microbiome analysis, we consider an experiment describing the change in gut microbiome for a single human patient who underwent a fecal microbiota transplant and antibiotic treatment using samples of a patient's fecal microbiome before and after antibiotic treatment, as studied in Taur et al.[8]. Procedure 1 details how to use the Kraken suite to analyze the overall diversity of microbiome samples while also statistically differentiating between microbiome compositions. Host-filtered microbial reads undergo classification against bacterial, viral, fungal, archaeal and human genomes using Kraken 2 (Fig. 1). The classification report is then used by Bracken for species abundance estimation, which provides estimated reads per species in the sample. Bracken output files are then passed to KrakenTools and Pavian for visualization. Pavian provides Sankey visualization of samples and read count comparisons between samples. For more in-depth downstream analysis, we recently developed KrakenTools (https://github.com/jenniferlu717/KrakenTools) as a set of individual programs for visualizing or transforming Kraken and Bracken output, available through GitHub or Bioconda (https://bioconda.github.io/recipes/krakentools/README.html). For microbiome projects, KrakenTools provides functions to compute Krona plots, filter and combine reports, and calculate α- and β-diversity metrics about each sample.

For pathogen identification, we consider a set of ten corneal samples from Li et al.[9]: eight separate human corneal samples with bacterial, viral or fungal infections and two non-infectious human corneal samples. Procedure 2 demonstrates how to identify pathogenic, atypical microbes within the overall microbiome. We classify the eight infected patient samples and two control samples against the same Kraken 2 database used in microbiome analysis (Fig. 1). However, the additional feature of unique *k*-mer counting from KrakenUniq[3] is enabled for accurate pathogen identification. We refer to this as Kraken2Uniq. Following classification by Kraken2Uniq, the Kraken report files are uploaded to the Pavian Shiny App[6], which compiles all of the read counts per species and allows between-sample visualization and comparison. The Pavian app then calculates *z*-scores between read counts, with higher *z*-scores resulting from samples with higher than usual read counts for a given species. We then sort the classified species by *z*-score, with the highest *z*-score species for each sample being the most likely pathogen. The filtered table can then be saved as a tab-delimited table that can undergo further visualization. Finally, pathogen identification can be validated using the `extract_kraken_reads.py` script from KrakenTools. The KrakenTools script allows users to extract classified reads and confirm potential pathogens using other tools such as BLAST[10] or Bowtie 2 (ref. [7]).

The target audience for this protocol is biologists and clinicians working in microbiome or metagenomics analysis. This protocol does not require programming expertise, but it does assume familiarity with the Unix command-line interface. Users should be comfortable running programs from the command line in the Unix environment.

Differences between Kraken methods

There are currently four major versions of the Kraken software: (1) Kraken[2], (2) KrakenUniq[3], (3) Kraken 2 (ref. [4]) and (4) Kraken2Uniq. All four versions are based on the same classification algorithm, which uses exact-matching of read *k*-mers against a Kraken database of *k*-mers from existing genomes. However, the methods differ in the ways they count, access and store the *k*-mer information, with each version improving upon the previous Kraken version.

Kraken and KrakenUniq apply the same *k*-mer retrieval strategy, and both tools can use the same databases. The reference database needs ~12 bytes per *k*-mer. Building and storing the Kraken index for a thousand distinct bacterial genomes of length 4 Mb would result in an index of size ~45 GB.

However, the RefSeq[11,12] database already contains ~64,000 bacterial species, and trying to store all *k*-mers from this ever-growing database became the main bottleneck of Kraken and KrakenUniq. Therefore, Kraken 2 was developed as a much more efficient version of Kraken, reducing the memory usage by 85% over Kraken and KrakenUniq. In addition to the reduction in database size, Kraken 2 also runs about five times faster than Kraken/KrakenUniq.

Another difference is in how the Kraken methods count reads. While Kraken and Kraken 2 provide cumulative statistics of the total read count per taxa, KrakenUniq and Kraken2Uniq additionally count and report the number of unique *k*-mers per taxon, using an efficient HyperLogLog implementation. In other words, for each species in the output, KrakenUniq will report how many

distinct *k*-mers from that species were observed in the reads. Unique counts are especially useful for pathogen detection to distinguish real from spurious signals.

Alternative analysis packages

The Kraken/KrakenUniq tools assign every read to a taxon of origin. Other software tools can also perform taxonomic classification, including CLARK[13], Centrifuge[14] and Kaiju[15]. Kraken 2 generally exhibits a more advantageous combination of speed, memory footprint and accuracy compared with those tools[4], but it is still possible to substitute those tools into the taxonomic classification step in this protocol (Step 2). Recent benchmarking studies[16,17] evaluated multiple metagenomic classifiers and found that Kraken, Kraken 2 and Bracken were among the best-performing methods.

Other tools seek to reduce the memory footprint of the index by including only the portions of the genome sequences that can distinguish between taxa, e.g., MetaPhlAn2 (ref. [18]). This reduces the size of the index, but prevents the tool from classifying many of the reads. So, while these tools can be used in a microbiome analysis like the one described in this protocol, they are not appropriate for pathogen identification.

While approaches based on large-scale machine learning have been shown to have advantages in certain scenarios, e.g., for classifying reads from species that are absent from the database[18], these approaches have tended to be computationally outperformed by classifiers that assign reads directly based on sequence content, without the need to first train a model[19,20]. For more details on competing paradigms for read classification, see Breitwieser et al.[21].

The Bowtie 2 tool used to filter host reads can be replaced with similar read alignment tools such as BWA-MEM[22] or minimap2 (ref. [23]).

Limitations

Removal of host reads

For metagenomic studies, removal of host (usually human) DNA is critical: either as a preprocessing step, using a program such as Bowtie 2 (ref. [7]) to align all of the reads to the host, or by including the host genome in the Kraken database. The preprocessing strategy may be more effective because external alignment programs such as Bowtie 2 are more sensitive than Kraken, and because preprocessing yields a greatly reduced set of reads, making subsequent steps faster. Bowtie 2 and other alignment programs are suitable for the preprocessing step because they can identify and remove any reads belonging to a known host (e.g., human), while Kraken can then identify the wide variety of unknown microbial species in the remainder of the sample. However, this preprocessing step may cause sequences of interest to be removed from the sample set, especially in the rare case of viral sequences found to be similar to a human endogenous retrovirus. In these cases, mapping the reads against a database of viral genomes without filtering human read beforehand might help to detect them[24]. In this protocol, we include the preprocessing with Bowtie 2 as an optional step that may or may not be included at the discretion of the users of the protocol.

Reference Database

Kraken 2's classification sensitivity and specificity highly depend on how (1) complete and (2) accurate the used reference database is. (1) The entire microbial biosphere is still far from being completely sequenced, and thus missing or partial genomes may introduce biases and reduce classification performance. Reads without a reference in the database will be labeled as unknown or imprecisely assigned to the next closest taxon. (2) Additionally, genomes can also be contaminated with DNA from other organisms[25,26] introduced during sequencing. Contamination causes Kraken to assign wrong taxonomic labels to reads. While this kind of classification error affects only a small fraction of reads, it can make the detection of weak signals difficult.

For the purpose of this protocol, we use the complete genomes from RefSeq bacteria, archaea, viral and plasmid libraries along with extensively screened eukaryotic pathogen genomes[27]. We use only complete genomes to avoid potential contamination from draft genomes.

Using the full Kraken 2 database requires 29 GB random-access memory (RAM). For users that may have limited RAM, Kraken 2 provides the ability to generate a minikraken database. Minikraken databases compress the full Kraken 2 database by saving a subset, but representative set, of the database *k*-mers. To allow all users to execute this protocol, we use an 8 GB minikraken database in the procedures.

Alternatively, Kraken 2 can also classify reads against protein databases using six-frame translation. Protein-level classification is more sensitive (but less specific) and therefore can help to reduce the missing reference bias. For even longer evolutionary distances, homology detection alignment-based classification methods such as DIAMOND[28] or MMseqs2 (ref. [29]) can be used.

Other issues can arise as reference databases grow and as the distribution of reference genomes per taxon becomes more lopsided. For example, Nasko et al.[30] showed that, as the RefSeq database has grown over time, Kraken has tended to assign a greater proportion of reads to higher levels of the taxonomy. In particular, they showed that a greater proportion of reads tend to be assigned to the genus level, at the expense of assignments to the species level and below. As public databases of reference genomes continue to grow, it will be important to continue to evaluate how classification results are affected by large or lopsided numbers of genomes per taxon.

Long read classification

This protocol is designed for Illumina sequencing reads, which are highly accurate (error rates <0.5%) but short, usually 100–250 bp. However, in some cases, users may want to evaluate longer, higher error-rate reads from Pacific Biosciences or Oxford Nanopore. For longer reads with a different error profile, users might remove host DNA using minimap2 (ref. [23]) rather than Bowtie2.

Also, the read count measurement might not be as useful for long reads, because these experiments typically generate fewer reads that differ greatly in length. For these experiments, it may make more sense to compare *k*-mer counts.

Assembly-based approaches

When a species is present in sufficiently high amounts, one might wish to assemble the reads de novo to create large, contiguous sequences from that species. The procedures described here do not perform genome assembly, which requires other software methods (for a recent review, see Yang et al.[31]).

Materials

Equipment

▲ **CRITICAL** The Kraken 2 protocol website http://ccb.jhu.edu/data/kraken2_protocol/ summarizes all required software and data, along with details about how to download all required data for the protocol.

Required software and hardware

- Kraken 2 software (https://github.com/DerrickWood/kraken2/, version 2.1.1 or later)
- Bracken software (https://github.com/jenniferlu717/Bracken, version 2.6.2 or later)
- KrakenTools software (https://github.com/jenniferlu717/KrakenTools, version 1.1. or later)
- Pavian software (https://github.com/fbreitwieser/pavian, version 1.0 or later)
- Bowtie 2 (https://github.com/BenLangmead/bowtie2, version 2.4.4)
- samtools (http://www.htslib.org/download/, version 0.1.20 or later)
- R (https://www.r-project.org)
- RStudio (https://www.rstudio.com/)
- Hardware (64-bit computer running either Linux or Mac OS X (10.7 Lion or later); 8 GB of RAM; see 'Equipment setup')

Required data

- The example corneal reads, microbiome fecal reads and Kraken 2 database for use in this protocol are available from NCBI SRA and Amazon AWS
- The microbiome analysis portion of this protocol is illustrated with a sample microbiome dataset of fecal samples from Taur et al.[8]. Many samples from the paper are available at https://www.ncbi.nlm.nih.gov/bioproject/PRJNA491657. Here we use three samples from a single patient, T11, who began antibiotic treatment at day 0, underwent stem cell engraftment on day 14 and received autologous fecal microbiota transplantation (auto-FMT) to return the patient's gut microbiota diversity at day 29
- The infectious disease analysis portion of this protocol is illustrated with selected corneal samples from Li et al. [9]. All 20 samples from the paper are available at https://www.ncbi.nlm.nih.gov/bioproject/PRJNA381365. However, for clarity, we used a select group of ten of the corneal samples (including two controls)

- Both procedures require a Kraken 2 database. For this protocol, we use the same prebuilt Kraken 2 database, containing RefSeq complete genomes from December 2020 for bacterial, archaeal and viral genomes, the human reference genome GRCh38 and the cleaned eukaryotic pathogen genomes for EuPathDB48 (ref. [27]) (available at http://ccb.jhu.edu/data/eupathDB/). Additional prebuilt Kraken 2 databases are available in the AWS cloud, detailed here: https://benlangmead.github.io/aws-indexes/k2. 'Equipment setup' details how to create a Kraken 2 database if the desired database is not already available

Equipment setup

Kraken 2 databases

Several prebuilt Kraken 2 databases are available at https://benlangmead.github.io/aws-indexes/k2. The most commonly used database is the standard Kraken 2 database (which includes RefSeq archaea, bacteria, viruses, plasmid complete genomes, UniVec Core and the most recent human reference genome, GRCh38). In addition to the standard database, we provide a database of the cleaned eukaryotic pathogens[27], 16S rRNA databases and expanded standard databases with RefSeq protozoa, fungi and plant genomes. All databases are updated monthly to include the most recent genomes.

In this protocol, we use an 8 GB minikraken database subset of the combined standard Kraken 2 database with the cleaned eukaryotic pathogen genomes.

If the desired database is not available, we describe here how to create a custom Kraken 2 database using the `kraken2-build` script options:

- First, download the NCBI taxonomy

```
$ kraken2-build --db krakendb --download-taxonomy
```

- Second, download one or more reference libraries. (The full list of available libraries is at https://github.com/DerrickWood/kraken2/wiki/Manual#custom-databases)

```
$ kraken2-build --db krakendb --download-library bacteria
$ kraken2-build --db krakendb --download-library archaea
$ kraken2-build --db krakendb --download-library viral
$ kraken2-build --db krakendb --download-library protozoa
$ kraken2-build --db krakendb --download-library UniVec_Core
```

- Third, download additional genomes by adding multi-FASTA or single-FASTA files. The FASTA sequence headers must include either (1) NCBI accession numbers or (2) the text `kraken:taxid` followed by the taxonomy ID for the genome (e.g., `>sequence100|kraken:taxid|9606|`). If this requirement is met, the following commands will add the sequences to the database:

```
$ kraken2-build --db krakendb --add-to-library chr1.fa
$ kraken2-build --db krakendb --add-to-library chr2.fa
```

- Finally, build the Kraken 2 database and generate the Bracken database files

```
$ kraken2-build --db krakendb --build --threads 8
$ bracken-build -d krakendb -t 8 -k 35 -l 100
```

Downloading and organizing required data

This section details how to organize the database and samples used in this protocol. You should first download the database from https://genome-idx.s3.amazonaws.com/kraken/k2_standard_eupath_20201202.tar.gz and the samples from NCBI SRA. For details on how to download the SRA samples, see https://www.ncbi.nlm.nih.gov/sra/docs/sradownload/. Box 1 lists the SRA accession IDs for each of the samples used in the Kraken microbiome analysis procedure and the Kraken pathogen identification procedure.

- Create four folders for required data:

```
k2protocol_db, m_samples, p_samples, b_index
```

Box 1 | NCBI SRA samples

Downloading and installing software using conda

Conda is an open-source package manager that helps to install software. All the software tools mentioned here are packaged in Bioconda[40], which is a particular repository ('channel') within conda. To install the software using conda, use the following command:

```
$ conda install -c conda-forge -c bioconda kraken 2 krakentools bracken \
r bowtie2 samtools
```

Pavian has no conda package. To run the protocol, there are two ways:

- Install it locally. Open the R console and use the following command:

```
$ R if (!require(remotes)) { install.packages ("remotes")
remotes::install_github("fbreitwieser/pavian")
```

- Alternatively, use the Pavian webserver (https://fbreitwieser.shinyapps.io/pavian/)

Downloading and installing software using binaries

Alternatively, all software can be also downloaded as binary using the following commands:

- Create a directory to store all of the executable programs used in this protocol (if none already exists):

```
$ mkdir $HOME/bin
```

- Add the above directory to your PATH environment variable: `$ export PATH=$HOME/bin:$PATH`
- To install Kraken 2, download the latest release from https://github.com/DerrickWood/kraken2/releases, unpack the Kraken 2 zip archive, `cd` to the unpacked directory and run the install script

```
$ unzip v2.1.1.zip
$ cd kraken2-2.1.1/
$./install_kraken 2.sh.
```

- To install Bracken, download Bracken version 2.6.2 from https://github.com/JenniferLu717/Bracken, unpack the Bracken zip archive, `cd` to the unpacked directory and run the install script

```
$ unzip v2.6.2.zip
$ cd Bracken-2.6.2/
$ sh install_bracken.sh
```

- To install KrakenTools, download Kraken 2 version 2.1.1 from https://github.com/JenniferLu717/KrakenTools and unpack the KrakenTools zip archive

```
$ unzip v1.2
```

- Copy the kraken executables to a directory in your PATH, and create symlinks for the Bracken and KrakenTools executables

```
$ cp kraken2-2.1.1/kraken2 $HOME/bin
$ cp kraken2-2.1.1/kraken2 -build $HOME/bin
$ cp kraken2-2.1.1/kraken2 -inspect $HOME/bin
$ ln -s Bracken-2.6.2/bracken $HOME/bin/bracken
$ ln -s Bracken-2.6.2/bracken -build $HOME/bin/bracken-build
$ ln -s KrakenTools $HOME/bin/KrakenTools
```

- Download Bowtie 2 version 2.4.4 (x86 64) from https://github.com/BenLangmead/bowtie2/releases/

```
$ unzip bowtie2 -2.4.4-linux-x86_64.zip
$ cp bowtie2-2.4.4-linux-x86_64/bowtie2* $HOME/bin
```

- To install Pavian, open RStudio or R and run the following:

```
if (!require(remotes)) { install.packages ("remotes") }
remotes::install_github("fbreitwieser/pavian")
```

- Inside the `k2protocol_db` folder, unpack the database:

```
$ tar -xzvf k2_standard_eupath_20201202.tar.gz
```

- Inside the `b_index` folder, download the `k2protocol_bowtie2indices.tgz` file from http://ccb.jhu.edu/data/kraken2_protocol/ and unpack the files:

```
$ tar -xzvf k2protocol_bowtie2indices.tgz
```

- Download the fastq files for SRA samples for the microbiome analysis and for the pathogen identification analysis. Move the three microbiome analysis files into `m_samples`, and move the ten pathogen identification samples into `p_samples`

Microbiome analysis

Sample	SRA ID	File size	Description	Expected diversity
1	SRR14143424	5.9 GB	Paired, day −2	Normal
2	SRR14092160	3.4 GB	Paired, day −9	Normal
3	SRR14092310	2.3 GB	Paired, day 12	Low
Pathogen identification				
Sample	SRA ID	File size	Case ID	Expected pathogen
1	SRR12486971	178.2 MB	Case 10	*Anncaliia algerae*
2	SRR12486972	318.5 MB	Case 9	*Aspergillus fumigatus*
3	SRR12486974	141 MB	Case 7	*Candida albicans*
4	SRR12486978	110.5 MB	Case 3	*Mycobacteroides chelonae*
5	SRR12486979	112.3 MB	Case 20	Control
6	SRR12486981	311.1 MB	Case 18	Control
7	SRR12486983	236.5 MB	Case 16	HSV 1
8	SRR12486988	351 MB	Case 11	*Acanthamoeba castellanii*
9	SRR12486989	228.2 MB	Case 2	*Streptococcus agalactiae*
10	SRR12486990	465.7 MB	Case 1	*Staphylococcus aureus*

Procedure 1: microbiome analysis

▲ **CRITICAL** We will explore and visualize the depletion of patient T11's microbiota diversity due to antibiotic treatment, using our microbiome analysis pipeline. In the study by Taur et al.[8], all the patients underwent allogeneic hematopoietic stem cell transplantation where antibiotic treatment is essential for optimal clinical outcomes. However, antibiotics deplete a patient's microbiome diversity. The authors showed that auto-FMT treatment restores the microbial diversity to a patient's gut microbiome. In this example, we explicitly detail three samples from patient T11 collected before and during antibiotic treatment. Samples 1 and 2 were taken before T11 began antibiotic treatment, and sample 3 was collected during treatment.

Remove host DNA ● Timing ~10 min for three samples

1 Depending on the source of the metagenomic sample, users may remove host DNA by aligning reads to the host genome. The provided sample set has already undergone removal of host (human) DNA. Remove human or host DNA from other samples using Bowtie 2 (ref. [7]) as follows.

```
$ bowtie2 -x b_index/GRCh38 -p 8 -1 paired_reads_1.fastq -2 paired_
reads_2.fastq \
--un-conc nonhuman_reads.fastq -S human_reads.sam
```

Classify microbiome samples using Kraken ● Timing ~10 min

2 Run Kraken 2 (ref. [4]) to classify all reads, providing each read with its taxonomy identification. Kraken 2 examines the *k*-mers within a query sequence and uses this information to query a database. In normal usage, the only required inputs are the database name, specified with `--db`, and the input FASTA file (or two files if the reads are paired). Below is the simplest example:

```
$ kraken2 --db $DBNAME seqs.fq
```

In our protocol, we specify the number of threads to improve the timing; with more threads the program will generally finish faster. With `--report` we specify the location and name of the `k2report` file and output the results of the run to a `kraken2` file with `> path/to/file`. The `--minimum-hit-groups` flag specifies the minimum number of 'hit groups' needed to make a classification call. Hit groups are overlapping *k*-mers sharing the same minimizer. Kraken 2 uses

minimizers to compress the input genomic sequences, thereby reducing storage memory needed and run time. In this example, we increase the minimum number of hit groups from the default two groups to three groups for increased accuracy. Lastly, the `--report-minimizer-data` flag reports minimizer and distinct minimizer count information in addition to the normal Kraken 2 report.

```
$ kraken2 --db k2protocol_db --threads 8 --report kreports/
SRR14143424.k2report \
--report-minimizer-data --minimum-hit-groups 3 samples/SRR14143424_1.
fastq \
samples/SRR14143424_2.fastq > kraken_outputs/SRR14143424.kraken2
$ kraken2 --db k2protocol_db --threads 8 --report kreports/SRR14092160.
k2report \
--report-minimizer-data --minimum-hit-groups 3 samples/SRR14092160_1.
fastq \
samples/SRR14092160_2.fastq > kraken_outputs/SRR14092160.kraken2
$ kraken2 --db k2protocol_db --threads 8 --report kreports/SRR14092310.
k2report \
--report-minimizer-data --minimum-hit-groups 3 samples/SRR14092310_1.
fastq \
samples/SRR14092310_2.fastq > kraken_outputs/SRR14092310.kraken2
```

Kraken 2 has two output files, a standard Kraken 2 output, `.kraken2` file, and a Kraken 2 report, `.k2report` file. The standard Kraken 2 file outputs lines containing five tab-delimited fields; (1) 'C'/'U' indicating classified or unclassified, (2) sequence ID, (3) taxonomy ID (0 if unclassified), (4) length of sequence in base pairs and (5) a space-delimited list indicating the lowest common ancestor mapping of each *k*-mer. The Kraken report file outputs lines containing six tab-delimited fields: (1) percentage of fragments covered by the clade rooted at this taxon, (2) number of fragments covered by the clade rooted at this taxon, (3) number of fragments assigned directly to this taxon, (4) rank code such as S for species, (5) NCBI taxonomic ID and (6) indented scientific name. This information and more can be found in the Kraken 2 manual.

? TROUBLESHOOTING

Run bracken for abundance estimation of microbiome samples ● Timing <1 min

3 Following classification, use Bracken[5] to estimate species abundance in each sample. Here is a generic Bracken invocation:

```
$ bracken -d kraken_database-i sample.k2report -r read_length \
-l taxonomic_level -t read_threshold -o sample.bracken -w sample.breport\
```

Below are the commands used to generate the abundance estimation in our example. Here we set the read length as the average read length in the dataset: 100 bp. We set the level for abundance estimation to species (with `-l S`), and with the `-t 10` we require ten reads before abundance estimation to perform re-estimation. This effectively removes any species with fewer than ten reads, thereby removing some noise from low-abundance species.

```
$ bracken -d k2protocol_db -i kreports/SRR14143424.k2report -r 100 -l S
-t 10 \
-o bracken_outputs/SRR14143424.bracken -w breports/SRR14143424.breport
$ bracken -d k2protocol_db -i kreports/SRR14092160.k2report -r 100 -l S
-t 10 \
-o bracken_outputs/SRR14092160.bracken -w breports/SRR14092160.breport
$ bracken -d k2protocol_db -i kreports/SRR14092310.k2report -r 100 -l S
-t 10 \
-o bracken_outputs/SRR14092310.bracken -w breports/SRR14092310.breport
```

? TROUBLESHOOTING

Calculate α-diversity ● Timing ~35 s

4 When trying to measure biodiversity, it is useful to quantify differences within and between ecosystems. Whittaker et al.[32] defined α-diversity as a measure that captures the average diversity within a particular ecosystem. There are different α-diversity values that approximate the average diversity by computing a sort of weighted average for species counts in a sample, for example, Simpson's Index includes species proportional abundances in its calculation. Using the KrakenTools `alpha_diversity.py` script, you can calculate Berger Parker's[33] (BP), Fisher's[34] (F), Simpson's[35] (Si), inverse Simpson's (ISi)[35] and Shannon's[36] (Sh) α-diversity for each sample after running Kraken 2 and Bracken. You can specify which α-diversity metric you want to output to terminal with the `-a` flag, as shown below. Compute these five different types of α-diversity for each of the three samples using the following commands.

```
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14143424.bracken -a BP
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14143424.bracken -a Sh
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14143424.bracken -a F
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14143424.bracken -a Si
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14143424.bracken -a ISi
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092160.bracken -a BP
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092160.bracken -a Sh
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092160.bracken -a F
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092160.bracken -a Si
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092160.bracken -a ISi
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092310.bracken -a BP
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092310.bracken -a Sh
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092310.bracken -a F
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092310.bracken -a Si
$ python KrakenTools/DiversityTools/alpha_diversity.py \
-f bracken_outputs/SRR14092310.bracken -a Isi
```

? TROUBLESHOOTING

Calculate β-diversity ● Timing ~5 s

5 β-Diversity is useful when trying to examine the change in species diversity between two or more ecosystems. Next, use the `beta_diversity.py` script to compute the Bray–Curtis[37] dissimilarity matrix, which will contain the pairwise dissimilarities among three microbiome samples. With this metric, an output of 0 means the two samples are exactly the same and 1 means they are maximally divergent.

```
$ python KrakenTools/DiversityTools/beta_diversity.py -i \
bracken_outputs/SRR14092160.bracken       bracken_outputs/SRR14092310.
bracken \
bracken_outputs/SRR14143424.bracken --type bracken
```

? TROUBLESHOOTING

Generate Krona plots ● Timing <1 min

6 Krona plots are multilayered pie charts frequently used in metagenomic visualization for viewing data in a phylogenetic hierarchy. Using the output from Bracken, we can make these plots to visualize the number of reads coming from different species, genera, etc. as percentages. Run `kreport2krona.py` to generate Krona plots[38] using the following commands.

```
$ python KrakenTools/kreport2krona.py -r breports/SRR14143424.breport \
-o b_krona_txt/SRR14143424.b.krona.txt --no-intermediate-ranks
$ KronaScripts/ktImportText b_krona_txt/SRR14143424.b.krona.txt \
-o krona_html/SRR14143424.krona.html
$ python KrakenTools/kreport2krona.py -r breports/SRR14092160.breport \
-o b_krona_txt/SRR14092160.b.krona.txt --no-intermediate-ranks
$ KronaScripts/ktImportText b_krona_txt/SRR14092160.b.krona.txt \
-o krona_html/SRR14092160.krona.html
$ python KrakenTools/kreport2krona.py -r breports/SRR14092310.breport \
-o b_krona_txt/SRR14092310.b.krona.txt --no-intermediate-ranks
$ KronaScripts/ktImportText b_krona_txt/SRR14092310.b.krona.txt \
-o krona_html/SRR14092310.krona.html
```

? TROUBLESHOOTING

Generate Pavian plots using the Shiny app ● Timing <1 min

7 Using the `breport` files from the Bracken run (Step 3), you can use Pavian to create the plots shown in Fig. 2a–c. Figure 3 shows the Pavian interface and the steps for creating classification networks. Open the Pavian Shiny app via https://fbreitwieser.shinyapps.io/pavian/.

8 Click 'Browse…', and upload the three separate SRR*.breport files.

9 Once files are uploaded, click 'Sample' to see the hierarchical classification visualization results.

10 Click on the drop-down list to select the sample that you are viewing.

11 (Optional) Choose 'Configure Sankey' to change what taxonomical ranks to display, number of taxa at each level, etc. There are ten different settings that can be changed to customize the plots.

12 Save the network by clicking 'Save Network'.

Procedure 2: pathogen identification

Remove human DNA using Bowtie 2 ● Timing ~30 min for ten samples

1 (Optional) Remove human DNA from the sample set. The provided sample set has already undergone removal of host (human) DNA using the steps shown next, so they do not need to be run again. We provide these command lines as a guide for how to remove human or host DNA from other samples using Bowtie 2 (ref. [7]).

```
$ bowtie2 -x bowtie_index/GRCh38 -p 8 -f -1 paired_reads_1.fastq \
-2 paired_reads_2.fastq --un-conc nonhuman_reads.fa -S human_reads.sam
```

Classify reads with Kraken2-Uniq ● Timing ~6min, 42 s for ten samples (8 GB)

2 Following removal of host DNA, the remaining reads undergo classification using Kraken2Uniq. The `--report-minimizer-data` flag forces Kraken 2 to provide unique *k*-mer counts per classification. Run Kraken2 using the commands below. We use eight threads for faster run times, and `--minimum-hit-groups 3` for increased classification precision (i.e., fewer false positives).

```
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486971.k2report \
--paired SRR12486971_1.fastq SRR12486971_2.fastq > SRR12486971.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486972.k2report \
--paired SRR12486972_1.fastq SRR12486972_2.fastq > SRR12486972.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486974.k2report
```

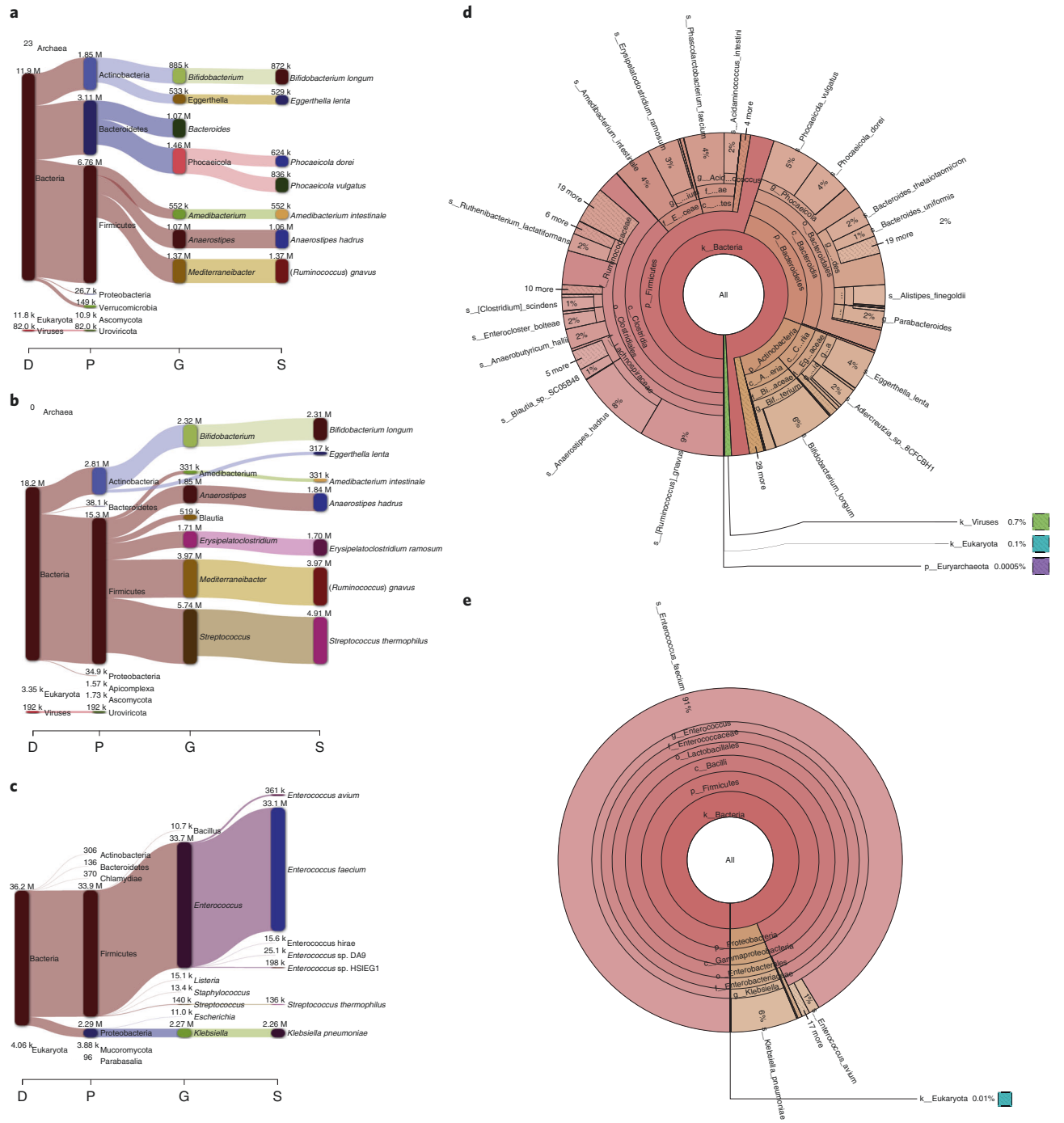

Fig. 2 | Microbiome plots. a–**c**, Pavian visualization for samples 1–3 from patient T11. Samples 1 and 2 have a similar taxonomic breakdown (**a** and **b**), corresponding to T11's normal microbiome diversity. **c** shows that sample 3 is dominated by a few bacteria while patient T11 is taking antibiotics. **d**,**e**, Krona plots generated from samples 2 and 3. Top: diversity of patient T11's microbiome before receiving any antibiotic treatment (sample 2). Bottom: depletion of his microbiome diversity while he is taking antibiotics (sample 3).

```
--paired SRR12486974_1.fastq SRR12486974_2.fastq > SRR12486974.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486978.k2report \
--paired SRR12486978_1.fastq SRR12486978_2.fastq > SRR12486978.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486979.k2report \
```

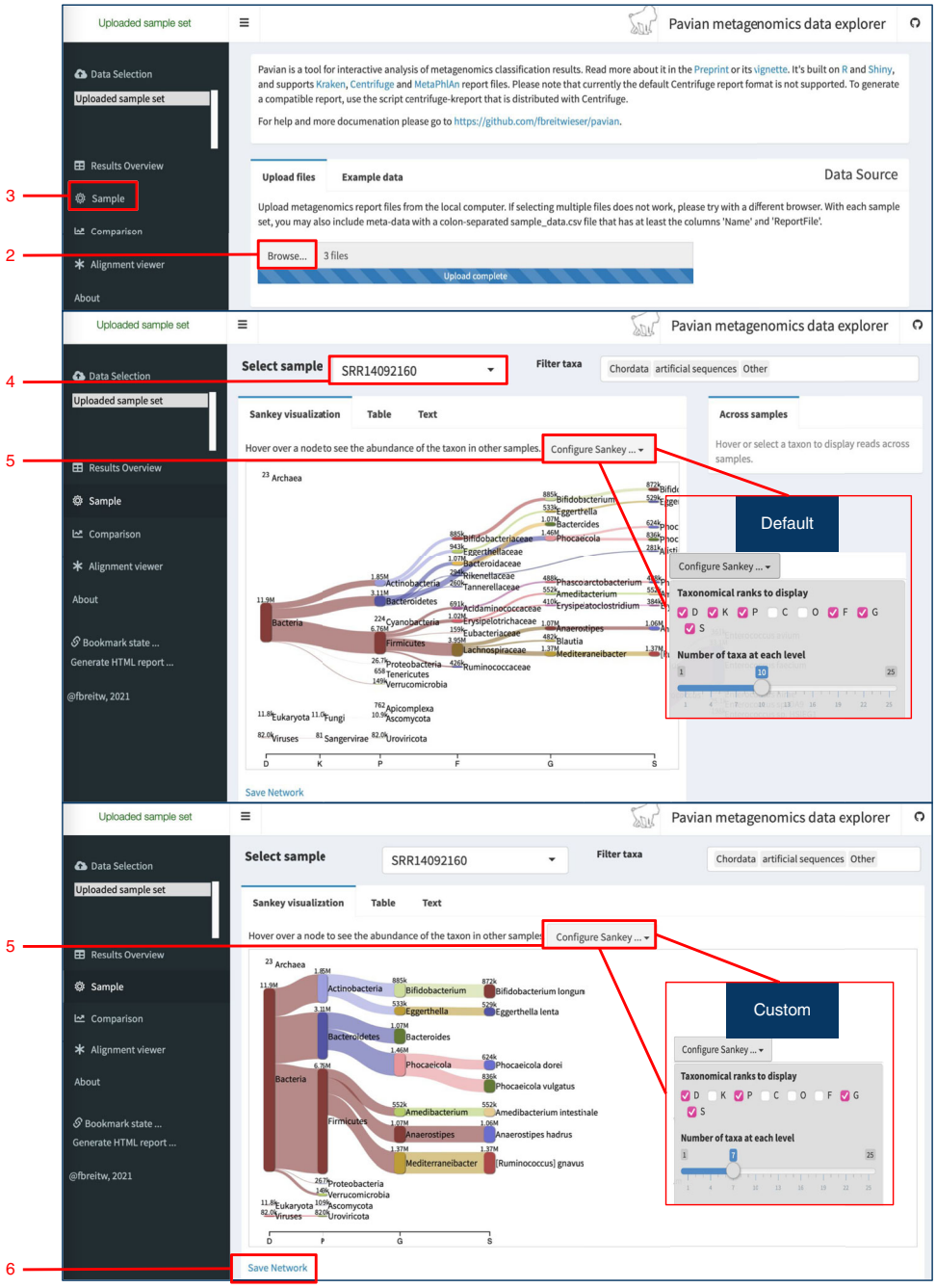

Fig. 3 | Pavian output for hierarchical visualization. Upon (1) opening the Pavian app, users should (2) upload the microbiome sample files. (3) Choose 'Sample' to view classification visualization results. (4) Select sample from the drop-down menu. (5) Select plot settings to customize visualization. (6) Save image of network.

```
--paired SRR12486979_1.fastq SRR12486979_2.fastq > SRR12486979.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486981.k2report \
--paired SRR12486981_1.fastq SRR12486981_2.fastq > SRR12486981.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486983.k2report \
--paired SRR12486983_1.fastq SRR12486983_2.fastq > SRR12486983.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486988.k2report \
--paired SRR12486988_1.fastq SRR12486988_2.fastq > SRR12486988.kraken2
```

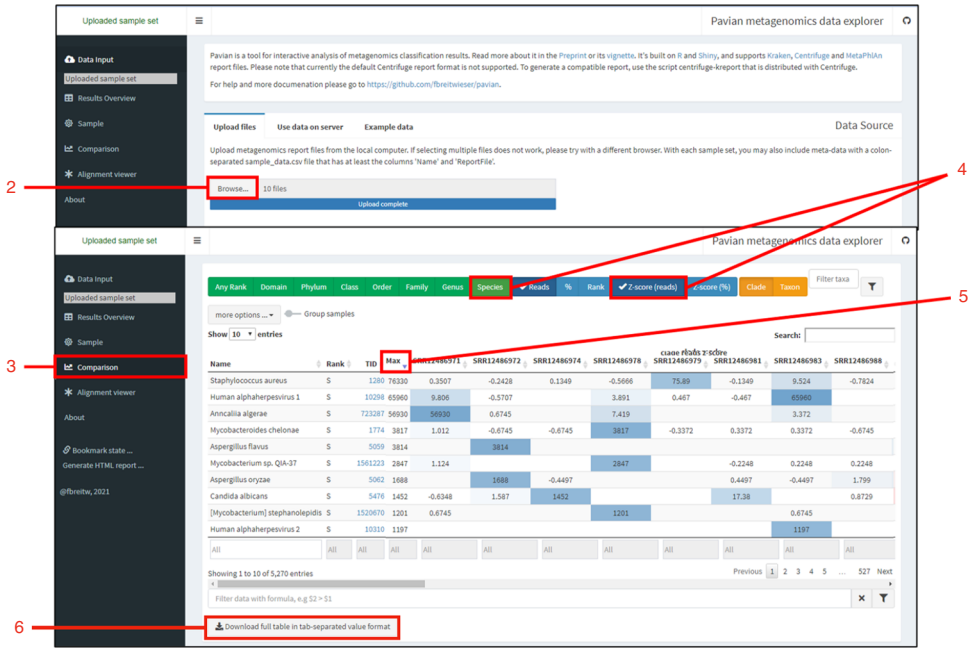

Fig. 4 | Pavian output for pathogen identification. Upon (1) opening the Pavian app, users should (2) upload the pathogen sample files. (3) Choose 'Comparison' to view the table of read counts per sample per taxon. (4) Select 'Species' and 'Z-score (reads)' to filter the table and calculate *z*-scores. (5). Finally, sort by maximum *z*-scores to focus on species that are most likely pathogen candidates.

```
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486989.k2report \
--paired SRR12486989_1.fastq SRR12486989_2.fastq > SRR12486989.kraken2
$ kraken2 --db k2protocol_db --threads 8 --minimum-hit-groups 3 \
--report-minimizer-data --report SRR12486990.k2report \
--paired SRR12486990_1.fastq SRR12486990_2.fastq > SRR12486990.kraken2
```

? TROUBLESHOOTING

Compare samples to controls using Pavian ● Timing ~5 min

3 Following classification, we compare samples using Pavian[6] (Fig. 4). Open the Pavian Shiny app via https://fbreitwieser.shinyapps.io/pavian/. Open the Pavian Shiny app via https://fbreitwieser.shinyapps.io/pavian/.

4 Click 'Browse…', and upload the ten separate SRR*.kreport2 files. Once files are uploaded, click 'Comparison' to see a table that summarizes the total read counts per organism in each sample.

5 Choose 'Species' and 'Z-score (reads)' to focus on species-level reads and calculate *z*-scores.

6 Sort by maximum *z*-scores.

▲CRITICAL STEP The species with the maximum *z*-score for each sample is the most likely pathogen. Note that if two or more pathogens have a high *z*-score in a given sample, we should consider the possibility that the infection has multiple causes. Species with high read counts (or high *z*-scores) across all samples are considered background noise or contamination.

Verify classification ● Timing ~30 min

7 For verifying the classification results, use the `extract_kraken_reads.py` script to extract reads for each of the top z-score species for each sample. This script extracts reads that matched a particular species, identified by the taxonomy ID that is provided with the `-t` parameter.

```
$ python extract_kraken_reads.py -k SRR12486971.kraken2 --include-
children \
-s SRR12486971_1.fastq -s2 SRR12486971_2.fastq -t 723287 \
-r SRR12486971.k2report -o SRR12486971_A.algerae.tid723287.1.fa \
```

```
-o2 SRR12486971_A.algerae.tid723287.2.fa
$ python extract_kraken_reads.py -k SRR12486972.kraken2 --include-
children \
-s SRR12486972_1.fastq -s2 SRR12486972_2.fastq -t 5059 \
-r SRR12486972.k2report -o SRR12486972_A.flavus.tid5059.1.fa \
-o2 SRR12486972_A.flavus.tid5059.2.fa
$ python extract_kraken_reads.py -k SRR12486974.kraken2 --include-
children \
-s SRR12486974_1.fastq -s2 SRR12486974_2.fastq -t 5476 \
-r SRR12486974.k2report -o SRR12486974_C.albicans.tid5476.1.fa \
-o2 SRR12486974_C.albicans.tid5476.2.fa
$ python extract_kraken_reads.py -k SRR12486978.kraken2 --include-
children \
-s SRR12486978_1.fastq -s2 SRR12486978_2.fastq -t 1774 \
-r SRR12486978.k2report -o SRR12486978_M.chelonae.tid1774.1.fa \
-o2 SRR12486978_M.chelonae.tid1774.2.fa
$ python extract_kraken_reads.py -k SRR12486983.kraken2 --include-
children \
-s SRR12486983_1.fastq -s2 SRR12486983_2.fastq -t 10298 \
-r SRR12486983.k2report -o SRR12486983_HSV1.tid10298.1.fa \
-o2 SRR12486983_HSV1.tid10298.2.fa
$ python extract_kraken_reads.py -k SRR12486988.kraken2 --include-
children \
-s SRR12486988_1.fastq -s2 SRR12486988_2.fastq -t 61605 \
-r SRR12486988.k2report -o SRR12486988_A.lugunensis.tid61605.1.fa \
-o2 SRR12486988_A.lugdunensis.tid61605.2.fa
$ python extract_kraken_reads.py -k SRR12486989.kraken2 --include-
children \
-s SRR12486989_1.fastq -s2 SRR12486989_2.fastq -t 1311 \
-r SRR12486989.k2report -o SRR12486989_S.agalactiae.tid1311.1.fa \
-o2 SRR12486989_S.agalactiae.tid1311.2.fa
$ python extract_kraken_reads.py -k SRR12486990.kraken --include-
children \
-s SRR12486990_1.fastq -s2 SRR12486990_2.fastq -t 1280 \
-r SRR12486990.k2report -o SRR12486990_S.aureus.tid1280.1.fa \
-o2 SRR12486990_S.aureus.tid1280.2.fa
```

8 Align extracted reads to the species genomes using alignment programs such as Bowtie 27 or Minimap2. We use Bowtie 2 for alignment.

```
$ bowtie2 -x b_index/Aalgerae -f -p 8 \
-1 SRR12486971_A.algerae.tid723287.1.fa \
-2 SRR12486971_A.algerae.tid723287.2.fa \
-S SRR12486971_A.algerae_aligned.sam
$ bowtie2 -x b_index/Aflavus -f -p 8 \
-1 SRR12486972_A.flavus.tid5059.1.fa \
-2 SRR12486972_A.flavus.tid5059.2.fa \
-S SRR12486972_A.flavus_aligned.sam
$ bowtie2 -x b_index/Calbicans -f -p 8 \
-1 SRR12486974_C.albicans.tid5476.1.fa \
-2 SRR12486974_C.albicans.tid5476.2.fa \
-S SRR12486974_C.albicans_aligned.sam
$ bowtie2 -x b_index/Mchelonae -f -p 8 \
-1 SRR12486978_M.chelonae.tid1774.1.fa \
-2 SRR12486978_M.chelonae.tid1774.2.fa \
-S SRR12486978_M.chelonae_aligned.sam
$ bowtie2 -x b_index/HSV1 -f -p 8 \
-1 SRR12486983_HSV1.tid10298.1.fa \
```

```
-2 SRR12486983_HSV1.tid10298.2.fa \
-S SRR12486983_HSV1_aligned.sam
$ bowtie2 -x b_index/Alug -f -p 8 \
-1 SRR12486988_A.lugunensis.tid61605.1.fa \
-2 SRR12486988_A.lugdunensis.tid61605.2.fa \
-S SRR12486988_Alug_aligned.sam
$ bowtie2 -x b_index/Sagalactiae -f -p 8 \
-1 SRR12486989_S.agalactiae.tid1311.1.fa \
-2 SRR12486989_S.agalactiae.tid1311.2.fa \
-S SRR12486989_Sagalactiae_aligned.sam
$ bowtie2 -x b_index/Saureus -f -p 8 \
-1 SRR12486990_S.aureus.tid1280.1.fa \
-2 SRR12486990_S.aureus.tid1280.2.fa \
-S SRR12486990_Saureus_aligned.sam
```

9 Following alignment, use SAMtools[39] to convert the SAM files to sorted BAM files.

```
$ samtools view -bS -F4 SRR12486971_A.algerae_aligned.sam \
> SRR12486971_A.algerae_aligned.bam
$ samtools view -bS -F4 SRR12486972_A.flavus_aligned.sam \
> SRR12486972_A.flavus_aligned.bam
$ samtools view -bS -F4 SRR12486974_C.albicans_aligned.sam \
> SRR12486974_C.albicans_aligned.bam
$ samtools view -bS -F4 SRR12486978_M.chelonae_aligned.sam \
> SRR12486978_M.chelonae_aligned.bam
$ samtools view -bS -F4 SRR12486983_HSV1_aligned.sam \
> SRR12486983_HSV1_aligned.bam
$ samtools view -bS -F4 SRR12486988_Alug_aligned.sam \
> SRR12486988_Alug_aligned.bam
$ samtools view -bS -F4 SRR12486989_Sagalactiae_aligned.sam \
> SRR12486989_Sagalactiae_aligned.bam
$ samtools view -bS -F4 SRR12486990_Saureus_aligned.sam \
> SRR12486990_Saureus_aligned.bam
$ samtools sort SRR12486971_A.algerae_aligned.bam \
-o SRR12486971_A.algerae_sorted.bam
$ samtools sort SRR12486972_A.flavus_aligned.bam \
-o SRR12486972_A.flavus_sorted.bam
$ samtools sort SRR12486974_C.albicans_aligned.bam \
-o SRR12486974_C.albicans_sorted.bam
$ samtools sort SRR12486978_M.chelonae_aligned.bam \
-o SRR12486978_M.chelonae_sorted.bam
$ samtools sort SRR12486983_HSV1_aligned.bam \
-o SRR12486983_HSV1_sorted.bam
$ samtools sort SRR12486988_Alug_aligned.bam \
-o SRR12486988_Alug_sorted.bam
$ samtools sort SRR12486989_Sagalactiae_aligned.bam \
-o SRR12486989_Sagalactiae_sorted.bam
$ samtools sort SRR12486990_Saureus_aligned.bam \
-o SRR12486990_Saureus_sorted.bam
$ samtools index SRR12486971_A.algerae_sorted.bam
$ samtools index SRR12486972_A.flavus_sorted.bam
$ samtools index SRR12486974_C.albicans_sorted.bam
$ samtools index SRR12486978_M.chelonae_sorted.bam
$ samtools index SRR12486983_HSV1_sorted.bam
$ samtools index SRR12486988_Alug_sorted.bam
$ samtools index SRR12486989_Sagalactiae_sorted.bam
$ samtools index SRR12486990_Saureus_sorted.bam
```

Fig. 5 | Pavian alignment viewer. a, The graphical interface of Pavian's alignment viewer. Users should upload the .bam and .bai files to the alignment viewer. **b**,**c**, Two example coverage plots for the pathogen identification samples. Pavian displays the coverage plot along with summarizing coverage statistics.

10 Finally, upload the sorted BAM file and the index of the sorted BAM file (.bam and .bai files) to Pavian's Alignment Viewer for the coverage plot and statistics. Figure 5 shows example coverage plots for the SRR12486978 reads aligned to the *Mycobacterium chelonae* genome and the SRR12486989 reads aligned to the *Streptococcus agalactiae* genome.

Troubleshooting

Step 2 (Procedure 1) and Step 2 (Procedure 2): run Kraken 2 results in large numbers of unclassified reads

If your Kraken 2 output files have too many unclassified reads, for example, 80% of unclassified reads, it probably means there are organisms missing from your working Kraken 2 database. To classify more reads, you should download additional genomes and add them to the database.

Steps 3–6 (Procedure 1): after Kraken 2 and Bracken calls

If there are any errors during these steps, you should check that your `kraken2` and `bracken` files were successfully made. For example, if the output from `kraken2` looks like this, there was a problem in the `kraken2` call:

```
Loading database information … done.
0 sequences (0.00 Mbp) processed in 0.001s (0.0 Kseq/m, 0.00 Mbp/m).
0 sequences classified (-nan %)
0 sequences unclassified (-nan %)
```

The command used to create the erroneous output shown above is as follows:

```
for sample in SRR14092160 SRR14092310; do
kraken2 --db k2protocol_db --threads 8 --report kreports/$sample.k2report \
--report-minimizer-data --minimum-hit-groups 3 samples/$sample_1.fasta \
samples/$sample_2.fasta > kraken_outputs/$sample.kraken2
done
```

The problem in this example is the bash syntax. Here, the input filenames are not interpolated correctly because the "`$sample`" variable is not protected against trailing underscores in the filenames. Therefore, bash is then attempting to reference a nonexistent variable. In this example, the commands that follow will result in an error because 0 sequences were classified and the `.kraken2` file is empty. To fix this, you should check that you are referencing the files correctly. The output of a successful Kraken 2 run for sample SRR14092160 is as follows (you can specify as many samples as you want, separated by spaces):

```
Loading database information … done.
73021500 sequences (8907.00 Mbp) processed in 154.050 s (28440.6 Kseq/m,
3469 Mbp/m).
11996432 sequences classified (16.43%)
61025068 sequences unclassified (83.57%)
```

There should be non-zero values for the number of sequences processed and for the number of reads classified. This can be done by protecting the code via some form of quoting. The following command uses double quotes around the filenames to protect the code and runs Step 2, classification of samples with Kraken 2:

```
for sample in SRR14092160 SRR14092310; do
kraken2 --db k2protocol_db --threads 8 --report " kreports/${sample}.
k2report" \
--report-minimizer-data --minimum-hit-groups 3 "samples/${sample}_1.
fasta" \
"samples/${sample}_2.fasta" > "kraken_outputs/${sample}.kraken2"
done
```

Timing

Procedure 1: microbiome analysis
Step 1, removal of human DNA (optional): 10 min
Step 2, classification of microbiome samples: 10 min
Step 3, abundance estimation: 1 min
Step 4, calculate α-diversity: <35 s for all five alphas for three samples
Step 5, calculate β-diversity: 5 s
Step 6, generate Krona plots: <1 min
Steps 7–12, generate Pavian plots using the Shiny app: <1 min
Procedure 2: pathogen identification
Step 1, removal of human DNA (optional): 30 min
Step 2, classify reads with Kraken2-Uniq: <7 min for ten samples

Steps 3–6, comparison of samples versus controls: 5 min
Steps 7–10, validation of classification: 30 min

Anticipated results

The accuracy of classification and abundance estimation is dependent on the quality and composition of the genomes used in the Kraken and Bracken databases. If the database is missing genomes present in the sequenced samples, there might be a high proportion of unclassified reads when using Kraken. If the database contains contaminated genomes (e.g., bacterial genomes that have bits of human sequence in them[25,26]), this will lead to false positives, where sequencing reads will be identified as incorrect species. Unclassified reads and misclassified reads will lead to additional problems downstream for Bracken, which will only report abundances for species identified by Kraken. To ensure a higher classification percentage, we recommend using a comprehensive Kraken database containing, at minimum, the human genome and bacterial, viral, archaeal, vector and eukaryotic pathogen genomes. For a low rate of false positives, we recommend using only complete or quality-controlled genomes in the building of the Kraken database, such as those labelled as complete and found in NCBI, RefSeq or another curated database.

In this protocol, we use a 'minikraken' version of this comprehensive database that includes the human genome, complete bacterial, viral and archaeal genomes from RefSeq, and a filtered eukaryotic pathogen database. The minikraken version of the comprehensive database contains sequences from all of the aforementioned genomes, but it downsamples the *k*-mers from each genome sufficiently to allow Kraken to run with only 8 GB of RAM. Kraken should still detect accurate bacterial, viral, archaeal and eukaryotic microbial signatures, but as we limit the number of sequences, we also expect a large fraction of unclassified reads.

Microbiome analysis

The three metagenomic sequencing samples that we use in this protocol illustrate the normal state of patient T11's microbiome before treatment and its depleted diversity state after antibiotic treatment. (The original study[8] also measured diversity after a fecal microbial transplant restored much of the microbiome.) Supplementary Table 1 lists the read counts per species in the final Bracken analysis of the three samples. In Fig. 6a, we can find the computed α-diversity outputs for each sample, for each α-diversity type currently available. We calculate these values using the abundance estimation calculation from Bracken using KrakenTools' α-diversity script. In samples 1 and 2, both taken when patient T11 had normal microbiome diversity, we expect to see a higher value for α-diversity than in sample 3, which was taken during antibiotic treatment. As expected, in Fig. 6a, we see lower values for all the α-diversity types for sample 3, except for BP α, which we expect to be close to 1 in a low-diversity sample. The original study reported the α-diversity values shown in the last row of the table in Fig. 6a. The authors specified that it is IS α-diversity; however, they did not include the exact tool or equations they used to calculate these values. The IS α value on row 4 of the table in Fig. 6a is the value we found using Bracken's abundance estimation and using the equations shown. For sample 2, we found a large discrepancy in the α-diversity published in the original paper (10.55) and our calculation (21.5). However, this can be explained by the large percentage of unclassified reads for this sample (~80%). Since the Kraken 2 report has many unclassified reads, the Bracken abundance estimation is not being calculated for the entire sample but only for the 20% of reads that are classified. To have more classified reads, a larger database needs to be used.

Figure 6b shows the output of the β-diversity calculation based on the Bray–Curtis dissimilarity matrix. The heat map shows that, when comparing the same two samples, for example day −2 (sample 1) with itself, the β-diversity value is 0 because the two samples being compared are exactly the same. When comparing day −2 with day 12 (sample 3), the β-diversity is close to 1 because the diversity levels in each are very different from each other. Also, as expected, we see that samples 1 and 2 (days −2 and −9) are somewhat similar with a β-diversity value of 0.6.

Shown in Fig. 2a–c are the Pavian plots for samples 1–3, respectively. Here we see that the plot in Fig. 2c is dominated by a single species of bacteria, *Enterococcus faecium*, showing that patient T11 had only one species of bacteria in his microbiome while undergoing antibiotic treatment. In Fig. 2a,b, we see several species of bacteria showing patient T11's normal variation in microbiome diversity. The Krona plot in Fig. 2d is divided into several different sections that represent the different species present in the sample, with no single species dominating. In the bottom Krona plot, (Fig. 2e), we see fewer partitions, showing again how *E. faecium* dominates this sample.

a

Alpha type	Sample 1	Sample 2	Sample 3	Equation
Shannon's	2.45	3.55	0.391	$\alpha_{shan} = -\Sigma_i^N p_i \log p_i$
Berger–Parker's	0.267	0.114	0.9137	$\alpha_{bp} = \max(p) = \frac{n_{max}}{N}$
Simpson's	0.845	0.953	0.161	$\alpha_{simp} = 1 - D$
Inverse Simpson's	6.48	21.5	1.19	$\alpha_{InvSimp} = \frac{1}{D}$
Fisher's	38.55	44.29	13.56	$S = \alpha_f \log(1 + \frac{N}{\alpha_f})$
Inverse Simpson's from FMT paper	8.596	10.55	1.05	Not given

b

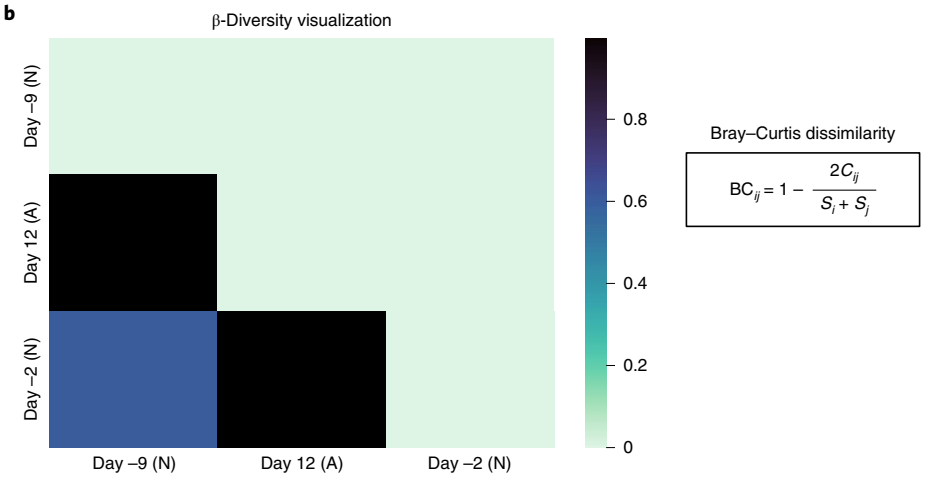

Fig. 6 | α- and β-diversity results. a, Computed results for α-diversity (D). In the equations, p is a vector of p_is where $p_i = \frac{(\text{number of individuals in } i^{\text{th}} \text{ species})}{\text{total number of individuals}}$ for all i species, i.e., $p_i = \frac{n_i}{N}$. And $D = \frac{\sum n_i(n_i-1)}{N \times (N-1)}$. **b**, Heatmap of the three samples from three different timepoints in patient T11's treatment. The sample taken on day 12 after starting antibiotic treatment (day 0) is marked as 'day 12 (A)' to denote that antibiotics were being used at the time of sample retrieval. The samples taken from 9 and 2 days before starting antibiotic treatment are marked as 'day −9 (N)' and 'day −2 (N)' respectively, to mark that they should have normal diversity (N). Here we use `beta_diversity.py` to compare diversity across samples. The Bray–Curtis dissimilarity formula (right) was used to calculate the β-diversity. In this formula, C_{ij} is the sum of the minimum count for each species between samples, S_i and S_j are the sum of species counts for samples i and j respectively. We see that the two samples taken before commencement of treatment are more similar to each other than either sample compared with the day 12 sample.

Table 1 | Classification read counts from fecal samples using the minikraken database

Sample	SRA ID	Total reads	Unclassified reads	Unclassified (%)	Classified reads	Classified (%)
1	SRR14143424	61,266,111	42,899,387	70.02	18,366,724	29.98
2	SRR14092160	73,021,500	61,025,068	83.57	11,996,432	16.43
3	SRR14092310	53,998,936	17,827,967	33.02	36,170,969	66.98

Using the minikraken database, there will be many unclassified reads; a brief summary of the output can be seen in Tables 1 and 2. For plotting purposes, we simply discarded all unclassified reads from the `kreport2krona.py` output when creating the plots shown in Fig. 2. All of the visualizations methods used, Krona plot, Pavian plot and β-diversity heatmap, show the great difference in microbiome diversity between patient T11 before and during antibiotic treatment. All of the visualization methods used, Krona plot, Pavian plot and β-diversity heatmap, show the great difference in microbiome diversity between patient T11 before and during antibiotic treatment.

Pathogen detection

As we are using the minikraken version of our comprehensive database, we expect a high proportion of unclassified reads across all infectious disease samples. Table 3 lists the total read counts per sample and the number of unclassified and classified reads per sample. However, because the minikraken

Table 2 | Fecal samples per-clade read counts

Sample	SRA ID	Bacteria	Archaea	Virus	Eukaryota
1	SRR14143424	18,151,841	46	192,138	3,594
2	SRR14092160	11,892,389	57	82,062	12,034
3	SRR14092310	35,990,070	8	9	4,467

Per-clade read counts for the major clades in these samples. Only select major clades are listed with read counts for clarity.

Table 3 | Classification read counts from corneal samples using the minikraken database

Sample	Total reads	Unclassified reads	Unclassified (%)	Classified reads	Classified (%)
SRR12486971	3,664,512	2,899,189	79.1	765,323	20.9
SRR12486972	7,594,644	7,285,624	95.9	309,020	4.1
SRR12486974	3,335,998	3,162,788	94.8	173,210	5.2
SRR12486978	2,625,249	2,496,107	95.1	129,142	4.9
SRR12486979	2,371,302	2,023,600	85.3	347,702	14.7
SRR12486981	6,730,160	6,093,775	90.5	636,385	9.5
SRR12486983	4,819,760	3,234,956	67.1	1,584,804	32.9
SRR12486988	8,369,736	8,122,882	97.1	246,854	2.9
SRR12486989	5,440,369	5,252,849	96.6	187,520	3.4
SRR12486990	9,402,750	6,619,669	70.4	2,783,081	29.6

Table 4 | Cornea samples per-clade read counts

Sample	Bacteria	Archaea	Virus	Fungi	Amoeba
SRR12486971	649,685	34	228	85,396	45
SRR12486972	150,302	10	50	31,602	54
SRR12486974	85,361	7	14	36,833	18
SRR12486978	91,448	2	763	87	3
SRR12486979	317,804	21	64	885	36
SRR12486981	465,581	51	157	2,082	110
SRR12486983	888,749	44	646,017	1,496	54
SRR12486988	80,095	12	698	325	5587
SRR12486989	97,912	7	37	256	83
SRR12486990	2,674,036	152	1,109	4,449	166

This table lists the per-clade read counts for the major clades in these samples. Only select major clades are listed with read counts for clarity.

database was selected from a comprehensive set of genomes, we still expect to see classifications across all major taxonomic clades (e.g., bacteria, archaea, viruses, fungi, etc.). Table 4 lists the read counts for major clades, showing 80,000 to 2.7 million bacterial reads per sample, along with varying numbers of reads from archaea, fungi and amoebae. Full species read counts for each sample are listed in Supplementary Table 2.

Despite the many clades detected in the corneal samples, the diagnostic challenge is determining the most likely pathogen(s) per sample. Samples SRR12486979 and SRR12486981 are control samples, providing a baseline for read counts from species that are likely to be contaminants or possibly non-infectious components of the samples. For each of the remaining patient samples, we are interested in any species with higher-than-average read counts as compared with the control samples or other corneal samples. For example, sample SRR12486971 has 84,000+ *Anncaliia algerae* reads.

All other samples have 12 *A. algerae* reads or fewer, making it appear that *A. algerae* is a likely pathogen in that sample (which indeed it is, as was confirmed in the original study[9]).

The easiest method for finding the most likely pathogen is uploading a set of similar samples to Pavian[6], as described in Step 3 of the pathogen identification procedure. Using the comparison tab, Pavian can calculate and sort the species by *z*-scores, a metric used to determine whether a species (or any other taxonomic clade) has significantly higher read counts in one sample as compared with the remaining other samples. Table 5 lists the species with the highest *z*-score in each sample. As shown in the table, the highest *z*-score correctly identified the true pathogen for each of the noncontrol samples.

The species with the highest *z*-score can be further checked by using the *k*-mer-counting feature in KrakenUniq[3] (which is also available as a parameter in Kraken 2). As described in the KrakenUniq paper[3], for most species in a metagenomics sample, each read will be a random sample from the genome, which means that each read will tend to contribute approximately $r–k$ distinct *k*-mers to the *k*-mer count, where r is the read length. However, if a bacterial genome is contaminated with a small

Table 5 | Species with the highest *z*-scores in cornea samples

Sample	True infection	*z*-Score species	Taxid	Reads	*z*-Score
SRR12486971	*Anncaliia algerae*	*Anncaliia algerae*	723287	84,409	56,930
SRR12486972	*Aspergillus flavus*	*Aspergillus flavus*	5059	3,814	3,814
SRR12486974	*Candida albicans*	*Candida albicans*	5476	36,609	1,452
SRR12486978	*Mycobacterium chelonae*	*Mycobacterium chelonae*	1774	11,320	3,817
SRR12486979	Control	–	–	–	–
SRR12486981	Control	–	–	–	–
SRR12486983	*HSV 1*	*HSV 1*	10298	635,691	65,960
SRR12486988	*Acanthamoeba*	*Acanthamoeba lugdunensis*	61605	1,004	338
SRR12486989	*Streptococcus agalactiae*	*Streptococcus agalactiae*	1311	797	797
SRR12486990	*Staphylococcus aureus*	*Staphylococcus aureus*	1280	1,414,661	76,330

Species with the highest *z*-scores for each of the noncontrol samples.

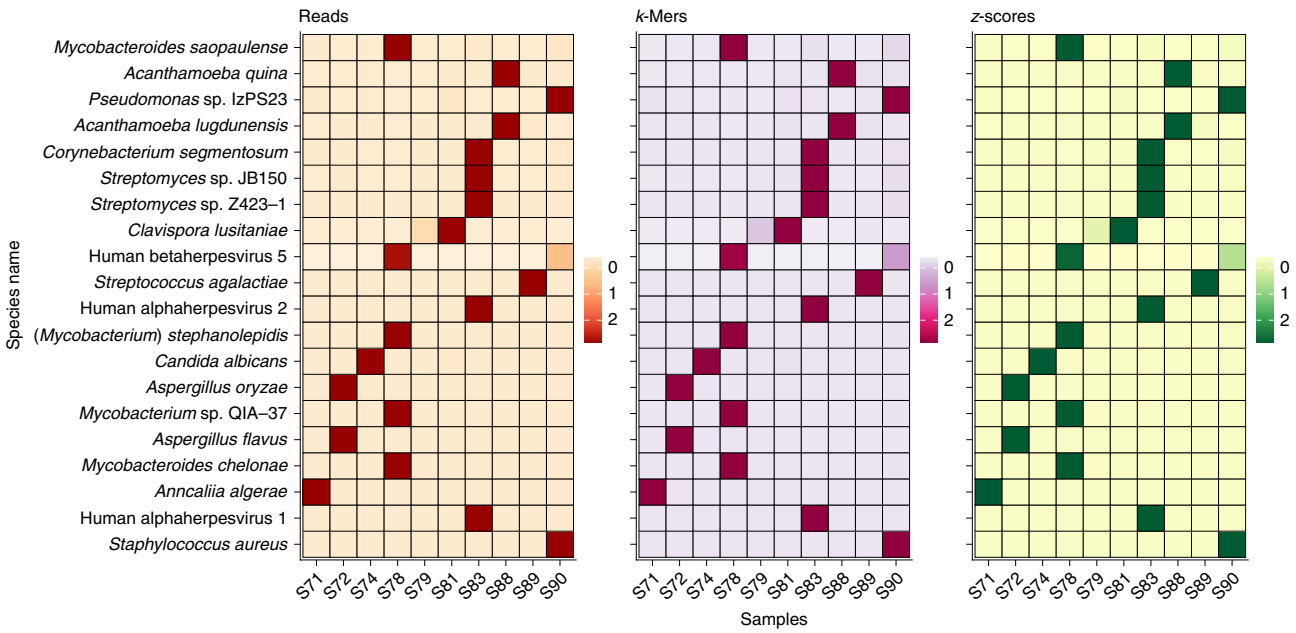

Fig. 7 | Pathogen identification results. The above plot summarizes the Kraken2Uniq results across the ten corneal samples. The number of reads, number of *k*-mers and *z*-scores reveal the most likely pathogen for each sample. For example, *Acanthamoeba quina* has high read and *k*-mer counts in S88 alone, *Staphylococcus aureus* is prevalent in S90, and human alphaherpesvirus 1 is likely to infect S83. For each sample, the true pathogen is the pathogen with the highest *z*-score for that particular sample.

amount of human sequence (for example), then large numbers of human reads will incorrectly match a small part of that genome. In these cases, the number of distinct *k*-mers that are observed from a genome will be very small compared with the number of reads. Thus, if the *k*-mer counts listed in the Kraken 2 report are relatively small, the species is probably not present but instead is a false positive caused by a contaminated genome in the database. Figure 7 summarizes the pathogen identification results for each individual sample, displaying a heat map of reads, *k*-mers and *z*-scores across all samples.

Finally, we validate the classification results by extracting the classified reads using KrakenTools and aligning the reads against the suspected pathogen genomes using Bowtie 2 (ref. [7]). We upload the alignment BAM files to Pavian[6] for a clear visualization of the read coverage across the genomes, confirming that the reads are derived from the full pathogen's genome.

As a result of following the pathogen identification procedure, we have discovered a set of potential infectious agents in the samples. The infectious agents should be presented to pathologists and clinicians for further verification with other independent methods.

Data availability

The microbiome analysis used three samples from Taur et al.[8], and the pathogen identification used ten samples from Li et al.[9], all of which can be found on NCBI with their SRA IDs. Source data are provided with this paper.

Code availability

The following website details and links all software and databases used in this protocol: http://ccb.jhu.edu/data/kraken2_protocol/. We also provide easy-to-use Jupyter notebooks for both workflows, which can be executed in the browser using Google Collab: https://github.com/martin-steinegger/kraken-protocol/.

References

1. Rappé, M. S. & Giovannoni, S. J. The uncultured microbial majority. *Annu. Rev. Microbiol.* **57**, 369–394 (2003).
2. Wood, D. E. & Salzberg, S. L. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* **15**, R46 (2014).
3. Breitwieser, F. P., Baker, D. N. & Salzberg, S. L. KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome Biol.* **19**, 198 (2018).
4. Wood, D. E., Lu, J. & Langmead, B. Improved metagenomic analysis with Kraken 2. *Genome Biol.* **20**, 257 (2019).
5. Lu, J., Breitwieser, F. P., Thielen, P. & Salzberg, S. L. Bracken: estimating species abundance in metagenomics data. *PeerJ Comput. Sci.* **3**, e104 (2017).
6. Breitwieser, P. & Salzberg, S. L. Pavian: interactive analysis of metagenomics data for microbiome studies and pathogen identification. *Bioinformatics* **36**, 1303–1304 (2020).
7. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012).
8. Taur, Y. et al. Reconstitution of the gut microbiota of antibiotic-treated patients by autologous fecal microbiota transplant. *Sci. Transl. Med.* **10**, eaap9489 (2018).
9. Li, Z. et al. Identifying corneal infections in formalin-fixed specimens using next generation sequencing. *Invest. Ophthalmol. Vis. Sci.* **59**(Jan), 280–288 (2018).
10. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**(Oct), 403–410 (1990).
11. Pruitt, K. D., Tatusova, T. & Maglott, D. R. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.* **35**, D61–D65 (2007).
12. O'Leary, N. A. et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.* **44**, D733–D745 (2016).
13. Ounit, R., Wanamaker, S., Close, T. J. & Lonardi, S. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative *k*-mers. *BMC Genomics* **16**, 236 (2015).
14. Kim, D., Song, L., Breitwieser, F. P. & Salzberg, S. L. Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Res.* **26**, 1721–1729 (2016).
15. Menzel, P., Ng, K. L. & Krogh, A. Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nat. Commun.* **7**, 11257 (2016).
16. Ye, S. H., Siddle, K. J., Park, D. J. & Sabeti, P. C. Benchmarking metagenomics tools for taxonomic classification. *Cell* **178**, 779–794 (2019).
17. Seppey, M., Manni, M. & Zdobnov, M. LEMMI: a continuous benchmarking platform for metagenomics classifiers. *Genome Res.* **30**, 1208–1216 (2020).

18. Segata, N. et al. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat. Methods* **9**, 811–814 (2012).
19. Vervier, K., Mahé, P., Tournoud, M., Veyrieras, J. B. & Vert, J. P. Large-scale machine learning for metagenomics sequence classification. *Bioinformatics* **32**, 1023–1032 (2016).
20. Luo, Y., Yu, Y. W., Zeng, J., Berger, B. & Peng, J. Metagenomic binning through low-density hashing. *Bioinformatics* **35**, 219–226 (2019).
21. Breitwieser, F. P., Lu, J. & Salzberg, S. L. A review of methods and databases for metagenomic classification and assembly. *Brief. Bioinform.* **20**, 1125–1136 (2017).
22. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1303.3997 (2013).
23. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094–3100 (2018).
24. Stephens, Z. et al. Exogene: a performant workflow for detecting viral integrations from paired-end next-generation sequencing data. *PLoS ONE* **16**, e0250915 (2021).
25. Breitwieser, F. P., Pertea, M., Zimin, A. V. & Salzberg, S. L. Human contamination in bacterial genomes has created thousands of spurious proteins. *Genome Res.* **29**, 954–960 (2019).
26. Steinegger, M. & Salzberg, S. L. Terminating contamination: large-scale search identifies more than 2,000,000 contaminated entries in GenBank. *Genome Biol.* **21**, 115 (2020).
27. Lu, J. & Salzberg, S. L. Removing contaminants from databases of draft genomes. *PLoS Comput. Biol.* **14**, e1006277 (2018).
28. Buchfink, B., Xie, C. & Huson, D. H. Fast and sensitive protein alignment using DIAMOND. *Nat. Methods* **12**, 59–60 (2015).
29. Mirdita, M., Steinegger, M., Breitwieser, F., Söding, J. & Levy Karin, E. Fast and sensitive taxonomic assignment to metagenomic contigs. *Bioinformatics* **37**, 3029–3031 (2021).
30. Nasko, D. J., Koren, S., Phillippy, A. M. & Treangen, T. J. RefSeq database growth influences the accuracy of *k*-mer-based lowest common ancestor species identification. *Genome Biol.* **19**, 165 (2018).
31. Yang, C. et al. A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data. *Comput. Struct. Biotechnol. J.* **19**, 6301–6314 (2021).
32. Whittaker, R. H. Evolution and measurement of species diversity. *Taxon* **21**, 213–251 (1972).
33. Berger, W. H. & Parker, F. L. Diversity of planktonic foraminifera in deep-sea sediments. *Science* **168**, 1345–1347 (1970).
34. Fisher, R. A., Corbet, A. S. & Williams, C. B. The relation between the number of species and the number of individuals in a random sample of an animal population. *J. Anim. Ecol.* **12**, 42–58 (1943).
35. Simpson, E. H. Measurement of diversity. *Nature* **163**, 688–688 (1949).
36. Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423 (1948).
37. Bray, J. R. & Curtis, J. T. An ordination of the upland forest communities of southern Wisconsin. *Ecol. Monogr.* **27**, 325–349 (1957).
38. Ondov, B. D., Bergman, N. H. & Phillippy, A. M. Interactive metagenomic visualization in a web browser. *BMC Bioinform.* **12**, 385 (2011).
39. Danecek, P. et al. Twelve years of SAMtools and BCFtools. *Gigascience* **10**, giab008 (2021).
40. Grüning, B. et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods* **15**, 475–476 (2018).

Acknowledgements

Indexes for tools in the Kraken suite, including the indexes used in this protocol, are made freely available on Amazon Web Services thanks to the AWS Public Dataset Program. B.L. was supported by NIH/NIHMS grant R35GM139602. S.L.S. was supported by NIH grants R35-GM130151 and R01-HG006677. M.S. acknowledges support from the National Research Foundation of Korea grant (2019R1A6A1A10073437, 2020M3A9G7103933, 2021R1C1C102065 and 2021M3A9I4021220); New Faculty Startup Fund; and the Creative-Pioneering Researchers Program through Seoul National University.

Author contributions

J.L. and M.S. led the development of the protocol. N.R. executed and designed the microbiome analysis protocol and is the author of the KrakenTools α-diversity tools. J.L. developed the pathogen identification protocol and is the author of Bracken and KrakenTools. M.S. authored the Jupyter notebooks for the protocol. D.E.W. is the senior author of Kraken and Kraken 2. F.B. is the author of KrakenUniq. C.P. is an author for the KrakenTools β-diversity script. B.L. supervised the development of Kraken 2. S.L.S. supervised the development of Kraken, KrakenUniq and Bracken. B.L. and S.L.S. supervised the development of this protocol. All authors contributed to the writing of the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-022-00738-y.

Correspondence and requests for materials should be addressed to Jennifer Lu or Martin Steinegger.

Peer review information *Nature Protocols* thanks the anonymous reviewers for their contribution to the peer review of this work.

Received: 29 June 2021; Accepted: 16 June 2022;
Published online: 28 September 2022

Related links

Key references using this protocol

Salzberg, S. et al. *Neurol. Neuroimmunol. Neuroinflamm.* **3**, e251 (2016): https://doi.org/10.1212/NXI.0000000000000251
Wood, D. et al. *Genome Biol.* **15**, R46 (2014): https://doi.org/10.1186/gb-2014-15-3-r46
Lu, J. et al. *Peer J. Comput. Sci.* **3**, e104 (2017): https://doi.org/10.7717/peerj-cs.104
Breitwieser, F. et al. *Genome Biol.* **19**, 198 (2018): https://doi.org/10.1186/s13059-018-1568-0
Wood, D. et al. *Genome Biol.* **20**, 257 (2019): https://doi.org/10.1186/s13059-019-1891-0
Breitwieser, F. et al. *Bioinformatics* **36**, 1303–1304 (2020): https://doi.org/10.1093/bioinformatics/btz715

Key data used in this protocol

Taur, Y. et al. *Sci. Transl. Med.* **10**, eaap9489 (2018): https://doi.org/10.1126/scitranslmed.aap9489
Li, Z. et al. *Invest. Ophthalmol. Vis. Sci.* **59**, 280–288 (2018): https://doi.org/10.1167/iovs.17-21617