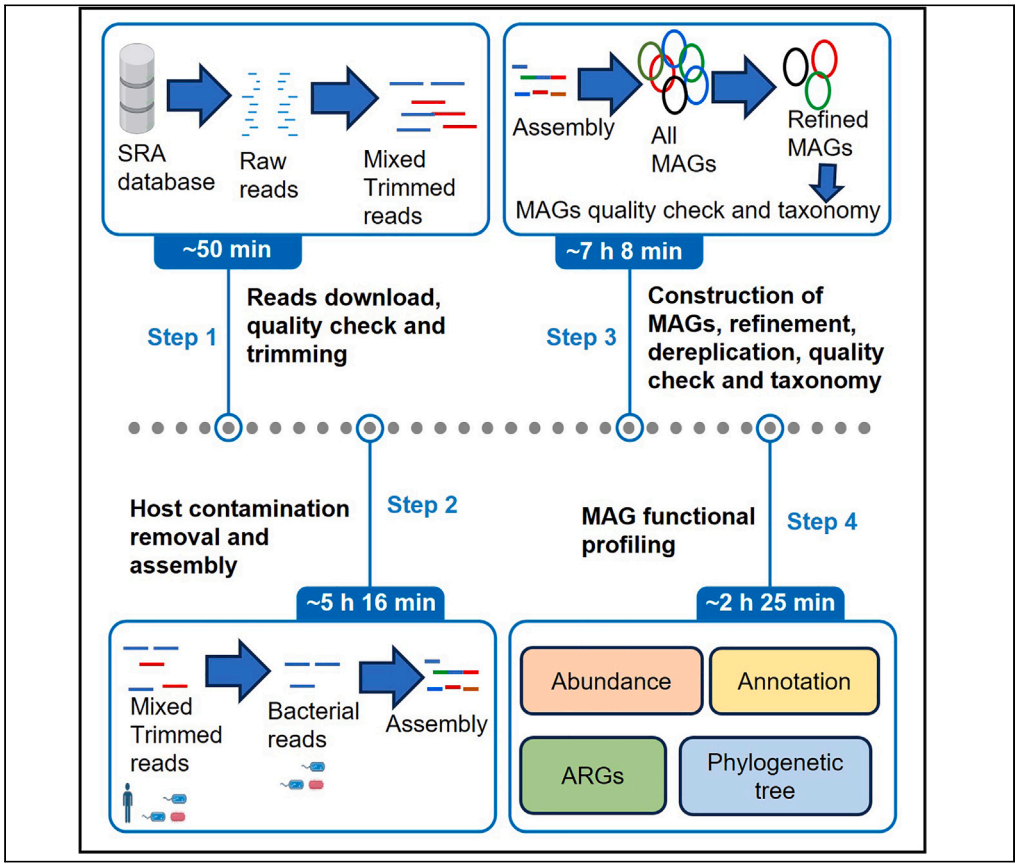
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for the construction and functional profiling of metagenome-assembled genomes for microbiome analyses



Constructing metagenome-assembled genomes (MAGs) from complex metagenomic samples involves a series of bioinformatics operations, each requiring deep bioinformatics knowledge. Here, we present a protocol for constructing MAGs and conducting functional profiling to address biological questions. We describe steps for system configuration, data downloads, read processing, removal of human DNA contamination, metagenomic assembly, and statistical quality assessment of the final assembly. Additionally, we detail procedures for the construction and refinement of MAGs, as well as the functional profiling of MAGs.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Goutam Banerjee, Suraya Rahman Papri, Pratik Banerjee

goutamb@illinois.edu (G.B.)
pratik@illinois.edu (P.B.)

Highlights

Step-by-step process of MAG construction from shotgun metagenomic sequencing data

MAG quality check, gene annotation, and taxonomy assignment

Functional profiling (such as abundance and antibiotic resistance genes) of MAGs

Banerjee et al., STAR Protocols 5, 103167
September 20, 2024
https://doi.org/10.1016/j.xpro.2024.103167

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for the construction and functional profiling of metagenome-assembled genomes for microbiome analyses

Goutam Banerjee,[1,2,*] Suraya Rahman Papri,[1] and Pratik Banerjee[1,3,*]

[1]Department of Food Science and Human Nutrition, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

[2]Technical contact

[3]Lead contact

*Correspondence: goutamb@illinois.edu (G.B.), pratik@illinois.edu (P.B.)
https://doi.org/10.1016/j.xpro.2024.103167

SUMMARY

Constructing metagenome-assembled genomes (MAGs) from complex metagenomic samples involves a series of bioinformatics operations, each requiring deep bioinformatics knowledge. Here, we present a protocol for constructing MAGs and conducting functional profiling to address biological questions. We describe steps for system configuration, data downloads, read processing, removal of human DNA contamination, metagenomic assembly, and statistical quality assessment of the final assembly. Additionally, we detail procedures for the construction and refinement of MAGs, as well as the functional profiling of MAGs.

BEFORE YOU BEGIN

Overview

Microbial communities and their abundance play crucial roles in various aspects, ranging from human health to soil health.[1–3] The culture-based methods have limitations in shedding light on the functional aspects of complex environments, such as the human gut.[4] The interactions of microbial communities in natural habitats are very complex; however, this complex consortium is important for completing various biological functions. Thus, metagenomic community-based analysis has gained significant attention owing to its enormous capability.[5] With the continuous advancements in high-throughput sequencing technologies, downstream bioinformatics analysis gain popularity for unraveling various functional aspects, including the construction of MAG, which might be able to provide insights into complex microbial communities and their functions.[6–9] It is worth noting that the concept of MAGs was first introduced by Tyson et al. in 2004, marking a significant milestone.[10] Since then, MAGs have been pivotal in providing valuable genomic information for taxonomically categorizing uncultured microorganisms, studying metabolic profiles, exploring microbiome dynamics, and investigating host-microbe interactions.[11] Additionally, it is also important to note that a crucial downstream application of MAG is their utilization for capturing variations in both sequence and microbiome structure.[12,13]

Several recent reports have been published emphasizing the importance of MAGs in several domains.[5,7,9] For example, MAGs offer researchers a powerful tool to delve into the vast diversity of microorganisms present in different environments and their roles in biogeochemical cycles,[14] disease processes,[15] and other ecological interactions.[16] The construction of MAGs is not a straightforward one-step process; it requires deep knowledge of bioinformatics. Moreover, emphasizing the functional profiling of MAGs adds an additional layer of complexity to the bioinformatics downstream pipelines. Usually, there are five mandatory steps involved in constructing MAGs: raw read

filtering and processing, metagenomic assembly, metagenomic binning, bin refinement, and bin taxonomy assignment. However, it is important to note that no standalone program can execute all these steps. For example, metagenomic assembly can be performed in MetaSPAdes,[17] MEGAHIT,[18] IDBA-UD,[19] etc. However, the selection of an assembly package is critical, as performance depends on several parameters, such as microbial community complexity, sequencing depth, and read length. Thus, the selection of the proper assembler requires prior bioinformatics experience and assembly statistics knowledge. Accordingly, we tried our best to cover the entire process, simplifying it from the initial raw read downloads to the comprehensive downstream processes involved in constructing MAGs through functional analysis.

Preparation: Prerequisite software installation

Timing: 1–2 h

1. Download conda and store it in the download folder.

```
1. Terminal-$: wget https://www.anaconda.com/download#downloads.
2. Terminal-$: cd download.
```

2. Install conda in Linux environment following the code.

```
1. Terminal-$: chmod +x Anaconda3-2023.09-0-Linux-x86_64.sh
2. Terminal-$: ./Anaconda3-2023.09-0-Linux-x86_64.sh
# Test installation
```

3. Check installation.

```
1. Terminal-$: conda list
```

Note: If the installation of conda is successful, you will see a list of packages displayed after entering the command 'conda list'. However, if the installation is unsuccessful, you will receive the message 'conda not found'. In such cases, please attempt to reinstall conda from the beginning. If the issue persists, refer to the conda documentation at 'https://docs.anaconda.com/free/troubleshooting/' or conda community 'https://community.anaconda.cloud/' for further assistance'.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data used		
SRR23604268	Zhang et al.[20]	https://www.ncbi.nlm.nih.gov/sra/?term=SRR23604268
SRR23604271	Zhang et al.[20]	https://www.ncbi.nlm.nih.gov/sra/?term=SRR23604272
Software and algorithms		
sra-tools v.3.0.8	NA	https://github.com/ncbi/sra-tools
fastp v.0.23.4	Chen et al.[21]	https://github.com/OpenGene/fastp

(*Continued on next page*)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
FASTQC v.0.12.1	Babraham[22]	https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
bowtie2 v.2.5.1	Langmead et al.[23]	https://github.com/BenLangmead/bowtie2
metaspades v.3.15.5	Nurk et al.[17]	https://github.com/ablab/spades
megahit v.1.2.9	Li et al.[18]	https://github.com/voutcn/megahit
metavelvet v.1.2.02	Namiki et al.[24]	https://github.com/hacchy/MetaVelvet
idba-ud v.1.1.3	Peng et al.[19]	https://github.com/loneknightpy/idba
Metaquast v.5.2.0	Mikheenko et al.[25]	https://github.com/ablab/quast
metawrap v.1.2	Uritskiy et al.[26]	https://github.com/bxlab/metaWRAP
drep v.3.4.5	Olm et al.[27]	https://github.com/MrOlm/drep
gtdbtk v.2.1.1	Chaumeil et al.[28]	https://github.com/Ecogenomics/GTDBTk
bwa v.0.7.17	Li et al.[29]	https://github.com/lh3/bwa
samtools v.1.18	Danecek et al.[30]	https://github.com/samtools/samtools
prokka v.1.14.6	Seemann[31]	https://github.com/tseemann/prokka
phylophlan v.3.0.3	Asnicar et al.[32]	https://github.com/biobakery/phylophlan
staramr v.0.10.0	Bharat et al.[33]	https://github.com/phac-nml/staramr
Others		
Human genome	NA	https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_000001405.26/
GTDB	Parks et al.[34]	https://gtdb.ecogenomic.org/downloads
Ubuntu v.20.04.1	NA	https://old-releases.ubuntu.com/releases/20.04.1/
conda v.23.7.4	NA	https://docs.anaconda.com/free/anaconda/release-notes/

NA: Not available.

MATERIALS AND EQUIPMENT

- Hardware: minimum 32 GB RAM and 1.5 TB storage required. The system runs on Ubuntu v.20.04.1, with x86_64 architecture, 40 CPUs, and conda v.23.7.4.
- Software used in this protocol is given in 'key resources table' (See software and algorithms section).
- We have used two publicly available SRA data sets (SRR23604268 and SRR23604271) for this tutorial which can be found under the bioproject number PRJNA938144.[20]

STEP-BY-STEP METHOD DETAILS

MAG construction, quality assessment, taxonomy assignment, and functional profiling are long processes, and thus, we have divided the entire process into four categories: basic protocol 1, basic protocol 2, basic protocol 3, and basic protocol 4.

Part 1. Conda environment setup, data download, quality check, and trimming (Basic protocol 1)

Timing: ~50 min

This section covers system configuration, data downloads, and read-processing steps. Conda is an open-source platform that simplifies environment management and installation of various packages. All the bioinformatics work was conducted within the Conda platform, leveraging its benefits, such as package compatibility, streamlined environment setup, version control, access to pre-built package repositories, and robust community support. Reads quality assessment and trimming of raw reads is the first step for any downstream bioinformatics analysis.

1. Conda environment setup (5 min).

a. Create conda environment named 'Basic_protocol_1':

```
1. (base) : conda create -n Basic_protocol_1
2. (base) :conda activate Basic_protocol_1
3. (Basic_protocol_1) :conda config –add channels bioconda
4. (Basic_protocol_1) :conda config –add channels conda-forge
```

2. Install sra-tools v.3.0.8 and fastp v.0.23.4 (5 min).[21]
   a. Install required packages with version:

```
1. (Basic_protocol_1) :conda install -c bioconda sra-tools==3.0.8 fastp==0.23.4
```

3. Download of raw reads from SRA database (20 min).
   a. Make a directory named 'MAG' and enter into this directory:

```
1. (Basic_protocol_1) :mkdir MAG
2. (Basic_protocol_1) :cd MAG
```

   b. Downloads SRA reads and generates paired end fastq files (SRA use 6 threads by default). If problem arise during operation, please follow the link for troubleshooting 'https://hpc.nih.gov/apps/sratoolkit.html#troubl':

```
1. (Basic_protocol_1) :prefetch SRR23604271 SRR23604268
2. (Basic_protocol_1) :fasterq-dump SRR23604271 --split-files –skip-technical
```

4. Download FastQC v.0.12.1 for quality check (20 min).[22]
   a. Download FastQC from the developer server, Unzip, activate, and open:

```
1. (Basic_protocol_1) :wget https://www.bioinformatics.babraham.ac.uk/projects/download.html.
2. (Basic_protocol_1) :Unzip FASTQC v0.12.1.zip
3. (Basic_protocol_1) :chmod +x FastQC
4. (Basic_protocol_1) :FastQC/fastqc
```

5. Data processing to remove low-quality reads (10 min).
   a. Make a directory named 'processed_reads' and store all processed reads:

```
1. (Basic_protocol_1) :mkdir processed_reads
```

   b. Based on FastQC result, reads are filtered using fastp (fastp use 3 threads by default):

```
1. (Basic_protocol_1) :fastp -i SRR23604268_1.fastq -I SRR23604268_2.fastq \
-o processed_reads /SRR23604268_R1.fastq -O processed_reads/SRR23604268_R2.fastq \
> -l 36 -q 30
```

```
2. (Basic_protocol_1):fastp -i SRR23604271_1.fastq -I SRR23604271_2.fastq \
-o processed_reads /SRR23604271_R1.fastq -O processed_reads /SRR23604271_R2.fastq \
> -l 36 -q 30
```

⚠ CRITICAL: Read quality check and subsequent processing is a very critical step that may impact downstream metagenomic data analysis, including MAG construction. So, very carefully read all parameters details from FastQC document (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/) before doing any trimming for your sample data. Here, we have used fastp with parameters: - l 36 (remove all reads below 36 bp), and -q 30 (quality score). In case of shallow reads (<10 million reads) user can set -q parameter to 20–25 to retain more reads for downstream analysis. For more details about the fastp parameters, please follow the GitHub repository page 'https://github.com/OpenGene/fastp'.

Note: At the end of this section, we will get the 4 read files (R1 and R2 for each sample) in the 'processed_reads' folder.

Part 2. Human DNA contamination removal, metagenomic assembly, and statistical quality assessment of final assembly (basic protocol 2)

Timing: ~5 h 16 min

Samples collected from hosts (like humans, mice, etc.) for microbiome study usually have host DNA contamination along with microbiome DNA, which may hinder the accuracy of downstream data analysis. To address this issue, we have used a mapping-based method to remove human DNA from metagenomic data selectively. The remaining microbial reads were used for the assembly process to reconstruct the genome sequences of individual microorganisms from fragmented DNA data. Nowadays, various assemblers have been introduced, each with its own set of strengths and weaknesses. However, their accuracy depends on several factors, and therefore, we used multiple assemblers to maximize our chances of obtaining a high-quality assembly. The final choice of assembler was selected based on the comprehensive statistics provided by metaQUAST.[25]

6. Conda environment setup (5 min).
   a. Deactivate previous environment and create new environment named "Basic_protocol_2":

```
1. (Basic_protocol_1):Conda deactivate
2. (base):conda create -n Basic_protocol_2
3. (base):conda activate Basic_protocol_2
4. (Basic_protocol_2):conda config –add channels bioconda
5. (Basic_protocol_2):conda config –add channels conda-forge
```

   b. Install required packages: bowtie2 v.2.5.1,[23] metaspades v.3.15.5,[17] megahit v.1.2.9,[18] metavelvet v.1.2.02,[24] idba-ud v.1.1.3,[19] Metaquast v 5.2.0[25]:

```
1. (Basic_protocol_2):conda install -c bioconda bowtie2==2.5.2 spades==3.15.5 \
megahit==1.2.9 metavelvet==1.2.02
```

7. Host contamination removal (90 min).
   a. Download the human genome from the NCBI database.
      Go to https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_000001405.26/ and download the fasta file named "ncbi_dataset.zip":
   b. Unzip the file in the terminal and copy the file 'CCA_000001405.15_GRch38_genomic.fna' to the MAG folder:

```
1. (Basic_protocol_2):unzip ncbi_dataset.zip

2. (Basic_protocol_2):cp

./ncbi_dataset/data/CCA_000001405.15/CCA_000001405.15_GRch38_genomic.fna ./MAG
```

   c. Rename the file CCA_000001405.15_GRch38_genomic.fna to reference_sequence.fasta.

```
1. (Basic_protocol_2):mv CCA_000001405.15_GRch38_genomic.fna reference_sequence.fna
```

   d. Making two directories:

```
1. (Basic_protocol_2):mkdir human_index clean_reads
```

   e. Indexing and human DNA contamination removal (bowtie2 use 1 thread by default). For any operational issue please read the manual 'https://gensoft.pasteur.fr/docs/bowtie2/2.1.0/':

```
1. (Basic_protocol_2):bowtie2-build reference_sequence.fasta \

human_index/reference_sequence.fasta.index

2. (Basic_protocol_2):bowtie2 -x human_index/reference_sequence.fasta.index -p 20 \

-1 processed_reads/SRR23604268_R1.fastq -2 processed_reads/SRR23604268_R2.fastq \

> --un-conc clean_reads/SRR23604268_reads.fastq -S human_reads.sam

3. (Basic_protocol_2):bowtie2 -x human_index/reference_sequence.fasta.index -p 20 \

-1 processed_reads/SRR23604271_R1.fastq -2 processed_reads/SRR23604271_R2.fastq \

> --un-conc clean_reads/SRR23604271_reads.fastq -S human_reads.sam
```

   f. Create the individual directory for each assembler and subdirectory for each sample and go back to the parent folder 'MAG':

```
1. (Basic_protocol_2):mkdir Assembly

2. (Basic_protocol_2):cd Assembly

3. (Basic_protocol_2):mkdir metaspades megahit metavelvet idba_ud

4. (Basic_protocol_2):cd ..
```

8. Metagenomic Assembly (185 min).
   a. Metaspades assembly: It will assemble fragmented reads into a long contig file (40 min) (metaspades use 16 threads by default). For any operational issue please read the manual 'https://home.cc.umanitoba.ca/~psgendb/doc/spades/manual.html':

```
1. (Basic_protocol_2):spades.py --meta \

-1 clean_reads/SRR23604268_R1.fastq -2 clean_reads/SRR23604268_R2.fastq \
```

```
-o Assembly/metaspades/Assembly_SRR23604268 --threads 8 --only-assembler
2. (Basic_protocol_2):spades.py --meta \
-1 clean_reads/SRR23604271_R1.fastq -2 clean_reads/SRR23604271_R2.fastq -o \
Assembly/metaspades/Assembly_SRR23604271 --threads 8 --only-assembler
```

b. Megahit assembly: It will assemble fragmented reads into a long contig file (45 min) (megahit use all threads by default). For any operational issue please read the manual 'https://www.metagenomics.wiki/tools/assembly/megahit':

```
1. (Basic_protocol_2):megahit \
-1 clean_reads/SRR23604268_R1.fastq -2 clean_reads/SRR23604268_R2.fastq \
-o Assembly/megahit/Assembly_SRR23604268
2. (Basic_protocol_2):Terminal-$: megahit \
-1 clean_reads/SRR23604271_R1.fastq -2 clean_reads/SRR23604271_R2.fastq \
-o Assembly/megahit/Assembly_SRR23604271
```

c. Metavelvet assembly: It will assemble fragmented reads into a long contig file (50 min) (metavelvet use all thread by default). For any operational issue please read the manual 'http://metavelvet.dna.bio.keio.ac.jp/MV.html':

```
1. (Basic_protocol_2):velveth Assembly/metavelvet/Assembly_SRR23604268 99 \
-short -separate -fastq \
> clean_reads/SRR23604268_R1.fastq clean_reads/SRR23604268_R2.fastq
2. (Basic_protocol_2):Terminal-$: velveth Assembly/metavelvet/Assembly_SRR23604271 99 \
-short -separate -fastq \
> clean_reads/SRR23604271_R1.fastq clean_reads/SRR23604271_R2.fastq
```

d. Idba-ud assembly: It will assemble fragmented reads into a long contig file (50 min) (idba-ud use all threads by default). For any operational issue please read the manual 'https://i.cs.hku.hk/~alse/hkubrg/projects/idba/index.html':

```
1. (Basic_protocol_2):fq2fa --merge clean_reads/SRR23604268_R1.fastq
clean_reads/SRR23604268_R2.fastq \
Assembly/idba_ud/SRR23604268_merge_IDBA.fa
2. (Basic_protocol_2):fq2fa --merge clean_reads/SRR23604271_R1.fastq
clean_reads/SRR23604271_R2.fastq \
Assembly/idba_ud/SRR23604271_merge_IDBA.fa
3. (Basic_protocol_2):idba_ud -r Assembly/idba_ud/SRR23604268_merge_IDBA.fa \
-o Assembly/idba_ud/Assembly_SRR23604268
4. (Basic_protocol_2):idba_ud -r Assembly/idba_ud/SRR23604271_merge_IDBA.fa \
-o Assembly/idba_ud/Assembly_SRR23604271
.
```

Note: Idba_UD takes single fasta file as input, and thus we have converted and merged both fastq files (R1 and R2) to a single fasta file using 'fq2fa' code.

9. Assembly statistics (30 min):

Note: This information will help in assessing the performance of the assembler which we have used during assembling steps. QUAST (statistical package) offers various parameters to measure assembling accuracy, key metrics such as N50 (indicating the length where half, or 50%, of the total base content is covered by contigs of that length or longer), contig numbers (total count of contigs in the assembly), and largest contig (length of the longest contig) are often regarded as pivotal. Usually, the higher the 'N50 values', lower the 'contig numbers', and the higher the 'largest contig length', the better. For more clarity, we have evaluated the statistics of two MAGs: MAG1 (completeness 91.2%) and MAG2 (completeness 100%). The comparative statistics are given below.

Parameters	MAG 1	MAG 2
N50	33898	82507
Contigs numbers	167	48
Largest contigs	127064	182676

So, based on the above parameters, MAG2 is better assembled compared to MAG1.

a. Download metaquast Python file from quast repository. For any operational issue please read the manual 'https://quast.sourceforge.net/docs/manual.html':

```
1. (Basic_protocol_2): git clone https://github.com/ablab/quast.
```

Note: 'metaquast.py' file will be within the folder 'quast'

b. Metaspades assembly statistics: (metaquast use 25% of available threads by default):

```
1. (Basic_protocol_2):quast/metaquast.py \
-o Assembly_statistics/metaspades/metaspades_SRR23604268 \
--threads 12 Assembly/metaspades/Assembly_SRR23604268/contigs.fa
2. (Basic_protocol_2):quast/metaquast.py \
-o Assembly_statistics/metaspades/metaspades_SRR23604271 \
--threads 12 Assembly/metaspades/Assembly_SRR23604271/contigs.fa
```

c. Megahit assembly statistic:

```
1. (Basic_protocol_2):quast/metaquast.py -o Assembly_statistics/megahit/megahit_SRR2360
4268 \
--threads 12 Assembly/megahit/Assembly _SRR23604268/final.contigs.fa
2. (Basic_protocol_2):quast/metaquast.py -o Assembly_statistics/megahit/megahit_SRR2360
4271 \
--threads 12 Assembly/megahit/Assembly_SRR23604271/final.contigs.fa
```

d. Metavelvet assembly statistic:

```
1. (Basic_protocol_2):quast/metaquast.py \
-o Assembly_statistics/metavelvet/metavelvet_SRR23604268 \
--threads 12 Assembly/metavelvet/Assembly_SRR23604268/contigs.fa
2. (Basic_protocol_2):quast/metaquast.py \
-o Assembly_statistics/metavelvet/metavelvet_SRR23604271 \
--threads 12 Assembly/metavelvet/Assembly_SRR23604271/contigs.fa
```

e. Idba_ud assembly statistic:

```
1. (Basic_protocol_2):quast/metaquast.py -o Assembly_statistics/idba_ud/idba_SRR2360
4268 \
--threads 12 Assembly/idba_ud/Assembly_SRR23604268/contig.fa
2. (Basic_protocol_2):quast/metaquast.py -o Assembly_statistics/idba_ud/idba_SRR2360
4271 \
--threads 12 Assembly/idba_ud/Assembly_SRR23604271/contig.fa
```

⚠ CRITICAL: Removal of host DNA contamination is essential to avoid analysis bias. Please check the updated version of the human genome file (if the updated version is available) for host contamination removal. Furthermore, metagenomic assembly depends on the assembly algorithm; thus, changes in the assembler may create variation in the expected result.

Note: Based on the summary statistics obtained from MetaQUAST, we have chosen the output generated by the 'metaspades' assembler for further downstream analysis. The assembly files (contigs.fa) for both samples can be located within the 'Assembly/metaspades' folder.

Part 3. Construction and refinement of metagenome-assembled genome along with taxonomy information (basic Protocol-3)

Timing: ~7 h 8 min

The construction and refinement of Metagenome-Assembled Genomes (MAGs) is a multistep process involving gathering individual genome information from metagenomic sequences and assembling them into nearly complete or complete individual genomes. Additionally, the quality of MAGs (assessed by completeness and contamination percentage) is very important. The detailed guidelines for MAG quality assessment and minimum information required for MAGs are given elsewhere.[35] This entire section deals with the construction of MAGs, several refinement steps (to increase completeness and reduce contaminations), and the taxonomy assignment of each MAG. Please note that MAGs with completeness above 90% and contamination below 5% were considered for this tutorial. Here we have used the metaspades assembled final assembly file (contigs.fa) as an input for both of these samples to construct MAGs.

10. Conda environment configuration (5 min).

a. Deactivate previous environment and create new environment named "Basic_protocol_3":

```
1. (Basic_protocol_2):conda deactivate
2. (base):conda create -n Basic_protocol_3
3. (base):conda activate Basic_protocol_3
4. (Basic_protocol_3):conda config –add channels bioconda
5. (Basic_protocol_3):conda config –add channels conda-forge
```

b. Install metawrap v.1.2,[26] drep v.3.4.5,[27] gtdbtk v.2.1.1[28]:

```
1. (Basic_protocol_3):conda install -c bioconda metawrap==1.2 drep==3.4.5 gtdbtk==2.1.1
```

11. Binning, bin-refinement, and bin-reassembled (320 min).
    a. Binning: (metawrap uses 1 thread by default). For any operational issue please read the manual 'https://readthedocs.org/projects/metawrap/downloads/pdf/v0.0.1/':

```
1. (Basic_protocol_3):metawrap binning -t 20 \
-a Assembly/metaspades/Assembly_SRR23604268/contigs.fa \
-o metaspades_SRR23604268_initial_binning --metabat2 --maxbin2 --concoct \
clean_reads/SRR23604268_R1.fastq clean_reads/SRR23604268_R2.fastq
2. (Basic_protocol_3):metawrap binning -t 20 \
-a Assembly/metaspades/Assembly_SRR23604271/contigs.fa \
-o metaspades_SRR23604271_initial_binning --metabat2 --maxbin2 --concoct \
clean_reads/SRR23604271_R1.fastq clean_reads/SRR23604271_R2.fastq
```

b. Bin refinement: This step will improve the bin quality in terms of completeness and contamination:

```
1. (Basic_protocol_3):metawrap bin_refinement \
-o metaspades_SRR23604268_bin_refinement -t 30 \
-A metaspades_SRR23604268_initial_binning/metabat2_bins/ \
-B metaspades_SRR23604268_initial_binning/maxbin2_bins/ \
-C metaspades_SRR23604268_initial_binning/concoct_bins/ -c 90 -x 5
2. (Basic_protocol_3):metawrap bin_refinement \
-o metaspades_SRR23604271_ bin_refinement -t 30 \
-A metaspades_SRR23604271_initial_binning/metabat2_bins/ \
-B metaspades_SRR23604271_initial_binning/maxbin2_bins/ \
-C metaspades_SRR23604271_initial_binning/concoct_bins/ -c 90 -x 5
```

c. Reassembled bins: In this step, bins were reassembled based on paired end read which will further enhance the completeness and reduce contamination:

```
1. (Basic_protocol_3):metawrap reassemble_bins -o SRR23604268_bin_reassembly \
-1 clean_reads/SRR23604268_R1.fastq -2 clean_reads/SRR23604268_R2.fastq -t 30 -m 800 \
-c 90 -x 5 -b metaspades_SRR23604268_bin_refinement/metawrap_90_5_bins
2. (Basic_protocol_3):metawrap reassemble_bins -o SRR23604271_ bin_reassembly \
-1 clean_reads/SRR23604271_R1.fastq -2 clean_reads/SRR23604271_R2.fastq -t 30 -m 800 \
-c 90 -x 5 -b metaspades_SRR23604271_bin_refinement/metawrap_90_5_bins
```

12. Dereplication of Bins (40 min).
    a. De-replicate all bins: (dRep use 6 threads by default). For any operational issue please read the manual 'https://drep.readthedocs.io/en/latest/index.html':

```
1. (Basic_protocol_3):dRep dereplicate dereplicated_SRR23604268 \
-g SRR23604268_bin_reassembly/reassembled_bins/*.fa
2. (Basic_protocol_3):dRep dereplicate dereplicated_SRR23604271 \
-g SRR23604271_ bin_reassembly/reassembled_bins/*.fa
```

13. Taxonomy of MAGs (60 min).
    a. Download the database using: 150 GB space required) (60 min):

```
1. (Basic_protocol_3):download-db.sh
```

    b. Path to database (please change the path of the env accordingly):

```
1. (Basic_protocol_3):conda env config vars set GTDBTK_DATA_PATH="/home/banerjee/
anaconda3/envs/gtdbtk-2.1.1/share/gtdbtk-2.1.1/db"
```

    c. Create a directory and store all MAG bins in this directory and changed the bin name manually to avoid conflict:

```
1. (Basic_protocol_3):mkdir MAG_taxonomy
2. (Basic_protocol_3):cp ./dereplicated_SRR23604268/dereplicated_genomes/*.fasta ./MAG_
taxonomy
3. (Basic_protocol_3):cp ./dereplicated_SRR23604271/dereplicated_genomes/*.fasta ./MAG_
taxonomy
```

    d. Bin taxonomy: (gtdbtk use cpus 1 by default). For any operational issue please read the manual 'https://ecogenomics.github.io/GTDBTk/index.html':

```
1. (Basic_protocol_3):gtdbtk classify_wf --genome_dir MAG_taxonomy -x fa --cpus 25 \
--genes --out_dir gtdb_result
```

e. Change the bin name with the respective taxonomy based on gttb result. Here we have shown an example of two bins from SRR23604268. Users can use the same code to change the names of all bins to their corresponding bacterial names.

```
1. (Basic_protocol_3): cd MAG_taxonomy

2. (Basic_protocol_3) MAG_taxonomy: mv SRR23604268.bin.1.fasta Bifidobacterium_longum.
fasta

3. (Basic_protocol_3) MAG_taxonomy: mv SRR23604268.bin.2.fasta Klebsiella_pneumoniae.
fasta
```

⚠ CRITICAL: During bin refinement, the selection of completeness and contamination percentage parameters are important. Here we have used > 90% completeness and < 5% contamination (-c 90 -x 5) to get high-quality MAGs. Adjusting these parameters may lead to obtaining a higher number of MAGs, encompassing both high and low quality. Please select the correct path to the database; otherwise, the GTDBTK function will not be executed properly.

Note: The binning results from the initial step can be located in the 'metaspades_SRR23604268_initial_binning' and 'metaspades_SRR23604271_initial_binning' folders. However, for the reassembled binning outcomes, you will find them in the 'SRR23604268_bin_reassembly/reassembled_bins' and 'SRR23604271_bin_reassembly/reassembled_bins' folders. Additionally, the results of bin-dereplication can be found in the 'dereplicate dereplicated_SRR23604268' and 'dereplicate dereplicated_SRR23604271' folders. At the end of this session, we can get all the MAGs in the 'MAG_taxonomy' folder.

Part 4. Functional profiling of MAG (basic protocol-4)

Timing: 2 h 25 min

This section covers a few important downstream bioinformatics analyses to predict some important functional information from generated MAGs, which is very important to understand its biological role in the community. The annotation step provides the genetic information (no. of total gene, CDS, tRNA, rRNA etc.) of MAGs. Based on their genetic distance matrix, the phylogenetic tree will place all the MAGs. We have not used any reference genome here, but users may include several references to build the tree. Additionally, the abundance calculation step will provide the relative abundance of each MAGs, which is essential for deducing the dominant genera in the sample. Understanding the antibiotic resistance profile in the genome is crucial for regulatory surveillance. Thus, we also have included this step to explore the suggestible and resistant bacterial genome along with detailed information like gene names and associated resistant antibiotics.

14. Conda environment configuration (5 min).
    a. Deactivate the previous environment and create a new environment named "Basic_protocol_4":

```
1. (Basic_protocol_3):conda deactivate

2. (base):conda create -n Basic_protocol_4

3. (base):conda activate Basic_protocol_4

4. (Basic_protocol_4):conda config --add channels bioconda

5. (Basic_protocol_4):conda config --add channels conda-forge
```

b. Install required packages: bwa v.0.7.17,[29] samtools v.1.18,[30] prokka v.1.14.6,[31] phylophlan v.3.0.3,[32] staramr v.0.10.0[33]:

```
1. (Basic_protocol_4):conda install -c bioconda bwa==0.7.17 samtools==1.18 \
prokka==1.14.6 phylophlan==3.0.3 staramr==0.10.0
```

15. Annotation of MAGs (40 min).
    a. Create a directory named 'MAG_function' and store all the MAG with taxonomy name:

```
1. (Basic_protocol_4):cd ..
2. (Basic_protocol_4):mkdir MAG_function
3. (Basic_protocol_4):cp ./MAG_taxonomy/*.fasta ./MAG_function
4. (Basic_protocol_4):cd MAG_function
```

    b. Annotation with prokka using loop: (prokka use 1 thread by default). For any operational issue please read the manual 'https://stab.standrews.ac.uk/wiki/index.php?title=Prokka&action=pdfbook&format=single':

```
1. (Basic_protocol_4):for F in *.fasta; do
N=$(basename $F .fasta) ;
prokka --locustag $N --kingdom Bacteria --addgenes --evalue 1e-9 --rfam --outdir $N --prefix $N $F ;
done
```

16. Abundance of MAGs (30 min): Here, we show only two MAGs, one from each sample, but you can follow the same code for all.
    a. Abundance of MAGs from sample SRR23604268 and SRR23604271: We utilize the MAGs as references to align the reads, in order to estimate the relative abundance of each MAG within the dataset. (bwa and samtools both use 1 thread by default). For any operational issue please read the manual BWA: 'https://www.animalgenome.org/bioinfo/resources/manuals/bwa.html' and samtools: 'https://www.htslib.org/doc/samtools.html':

```
1. (Basic_protocol_4):bwa index Bacteroides_uniformis.fasta
2. (Basic_protocol_4):bwa mem -t 30 MAG_function/Bacteroides_uniformis.fasta \
clean_reads/SRR23604268_R1.fastq clean_reads/SRR23604268_R2.fastq \
> Bacteroides_uniformis.sam
3. (Basic_protocol_4):samtools view -b MAG_function/Bacteroides_uniformis.sam \
> Bacteroides_uniformis.bam --threads 30
4. (Basic_protocol_4):samtools sort -o Bacteroides_uniformis.sorted.bam \
Bacteroides_uniformis.sam --threads 30
5. (Basic_protocol_4):samtools flagstat Bacteroides_uniformis.sorted.bam --threads 30
# Abundance of MAGs from sample SRR23604271
```

```
1. (Basic_protocol_4):bwa index Collinsella.fasta

2. (Basic_protocol_4):bwa mem -t 30 MAG_function/Collinsella.fasta \

clean_reads/SRR23604268_R1.fastq clean_reads/SRR23604268_R2.fastq \

> Collinsella.sam

3. (Basic_protocol_4):samtools view -b Collinsella.sam > Collinsella.bam --threads 30

4. (Basic_protocol_4):samtools sort -o Collinsella.sorted.bam Collinsella.sam –threads 30

5. (Basic_protocol_4):samtools flagstat Collinsella.sorted.bam --threads 30
```

17. Phylogenetic tree construction (30 min): Here, we have used the PhyloPhlAn to get a phylogenetic placement of the assembled MAGs. PhyloPhlAn 3 has the capability to automatically download the PhyloPhlAn database (180.5 Mb space required).[36] For any operational issue please read the manual 'https://huttenhower.sph.harvard.edu/phylophlan/'.

```
1. (Basic_protocol_4):cd ..
```

    a. Back to parent folder MAG:
    b. Configure file: To prepare the configure file please follow the instruction given in the developer page https://github.com/biobakery/biobakery/wiki/PhyloPhlAn-3:-Example-02:-Tree-of-life ':

```
1. (Basic_protocol_4):phylophlan_write_config_file \

  -d a \

  -o 02_tol.cfg \

  --db_aa diamond \

  --map_dna diamond \

  --map_aa diamond \

  --msa mafft \

  --trim trimal \

  --tree1 iqtree \

  --verbose 2>&1 | tee phylophlan_write_config_file.log
```

    c. Phylophlan tree generation: (Phylophlan use nproc 1 by default):

```
1. (Basic_protocol_4):phylophlan \

  -i MAG_function \

  -d phylophlan \

  -f 02_tol.cfg \

  --diversity low \

  -o pylophalan_result \

  --nproc 32 \

  --verbose 2>&1 | tee logs/phylophlan.log
```

18. ARG detection in MAGs (30 min).
    a. Back to parent folder MAG:

```
1. (Basic_protocol_4):cd ..
```

    b. Make a folder and copy all MAGs to this folder:

```
1. (Basic_protocol_4):mkdir AMR

2. (Basic_protocol_4):cp ./MAG_taxonomy ./AMR
```

    c. Update the ARG database: staramr use ResFinder database (space requirement 32.5 Mb):

```
1. (Basic_protocol_4):staramr db update –update-default
```

    d. Run staramr to predict ARGs: For any operational issue please read the manual 'https://github.com/phac-nml/staramr/blob/development/doc/tutorial/staramr-tutorial.ipynb':

```
1. (Basic_protocol_4):staramr search -o AMR_result AMR/*.fasta
```

⚠ CRITICAL: Functional profiling of MAG relies heavily on genome quality. Prioritizing high-quality MAGs, such as closed or nearly closed genomes, significantly enhances the accuracy of results in terms of functional annotation and metabolic pathway prediction. Moreover, it is essential to ensure that the ARG (Antibiotic Resistance Gene) database used for gene prediction is up-to-date. Incorporating the latest version of the ARG database improves the accuracy of gene prediction results in terms of gene coverage and identity percentage.

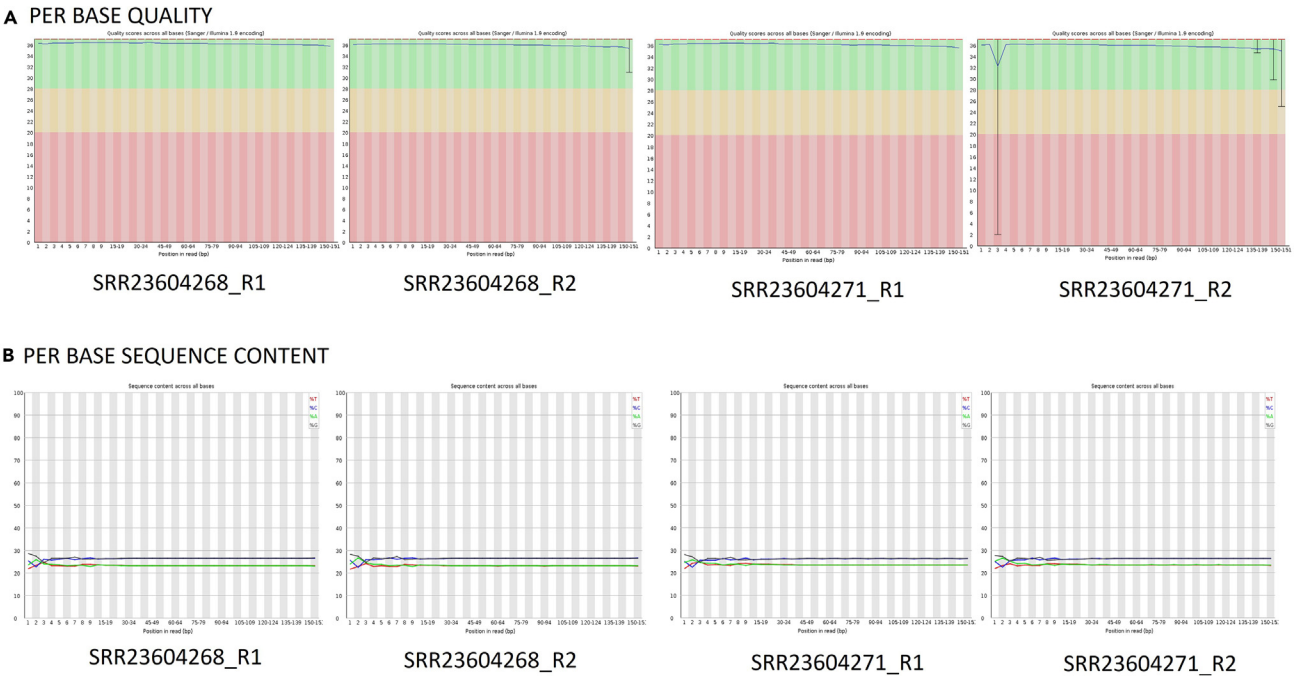


Figure 1. FastQC result of both samples after trimming of low-quality reads
(A) and (B) indicate per base quality and per base sequence content of both samples, respectively.

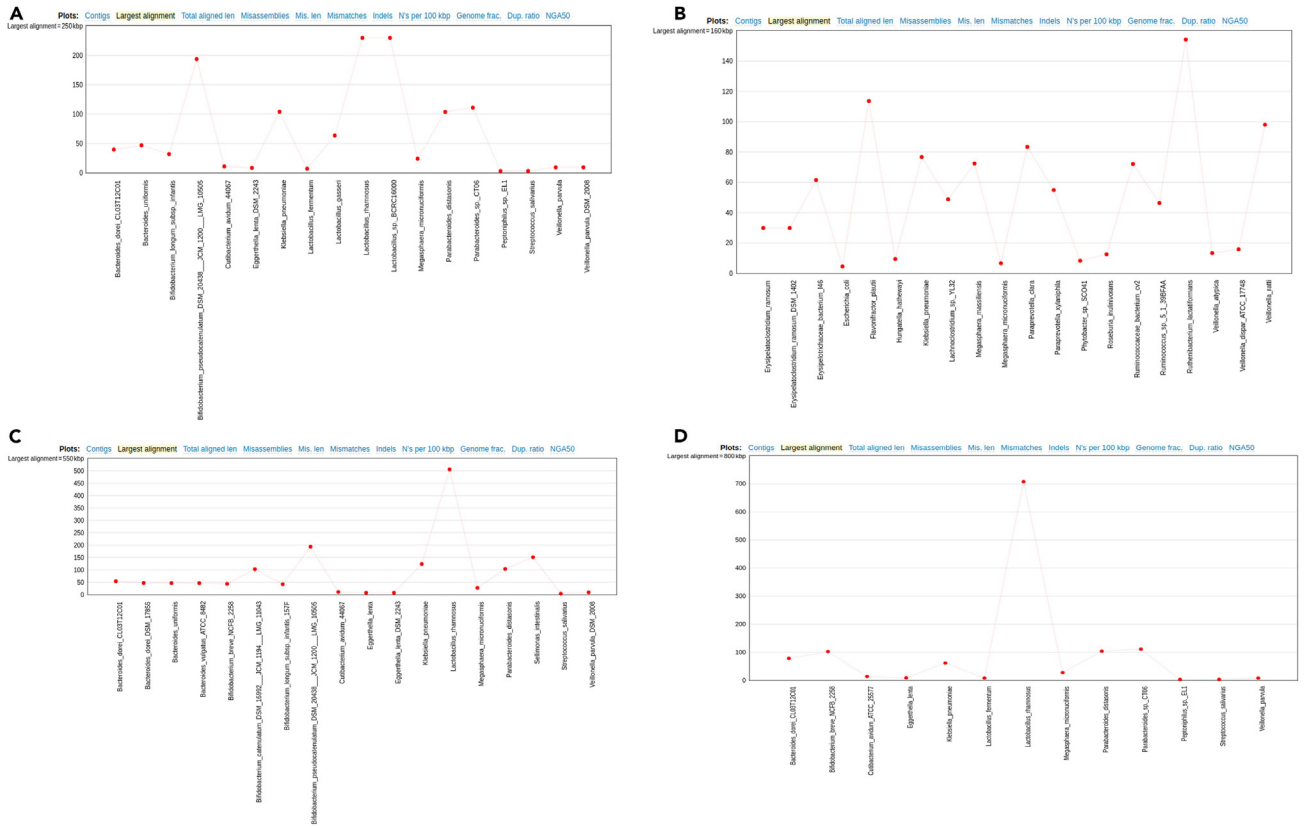


Figure 2. MetaQUAST statistics of final assembly file generated from sample SRR23604268

(A) indicates Idba_ud assembler, (B) megahit assembler, (C) metaspades, and (D) metavelvet.

Note: The annotation information of each MAG can be located in the folder named 'MAG_function'. At the same time, the AMR profile of each MAG can be obtained in 'AMR_result' folder.

EXPECTED OUTCOMES

The results obtained from this tutorial will provide clear and easily understandable insights. Within each section of the tutorial, we have thoughtfully included expected results at the end, ensuring that users can assess their progress effectively. In part 1 (Basic Protocol-1), the output consists of the FastQC results (Figures 1 and 2) for raw reads after trimming, with a particular focus on the percentage of reads retained (Table 1). For our tutorial, deep sequencing was employed, leading to the discarding of reads with a quality score (q) lower than 30. However, in scenarios involving shallow sequencing, a q < 15 threshold should be considered to retain a significant number of reads after

Table 1. Statistical summary of reads after filtering step using fastp package

Parameters	Sample SRR23604268				Sample SRR23604271			
Read	Read 1 before filtering	Read 1 after filtering	Read 2 before filtering	Read 2 after filtering	Read 1 before filtering	Read 1 after filtering	Read 2 before filtering	Read 2 after filtering
Total reads	37528127	35570663	37528127	35570663	39231247	37056105	39231247	37056105
Total bases	5666747177	5366469135	5666747177	5366469135	5923918297	5592805727	5923918297	5592805727
Q20 bases	5524453265 (97.489%)	5276018458 (98.3145%)	5440566369 (96.0086%)	5238644722 (97.6181%)	5773299980 (97.4575%)	5496133056 (98.2715%)	5671835814 (95.7447%)	5452566578 (97.4925%)
Q30 bases	5286581425 (93.2913%)	5078288900 (94.63%)	5141867121 (90.7375%)	4986225255 (92.9144%)	5518067113 (93.1489%	5285041160 (94.4971%)	5346529605 (90.2533%)	5178620352 (92.5943%)

Table 2. Accuracy comparison of four popular assembler packages assessed by MetaQUAST

Parameters	Sample SRR23604268				Sample SRR23604271			
Assembler	IDBA-UD	MetaVelvet	MetaSPAdes	MEGAHIT	IDBA-UD	MetaVelvet	MetaSPAdes	MEGAHIT
N50	9669	2800	20672	15473	12991	6373	23312	18918
N90	865	626	1044	1032	877	660	823	874
Contigs numbers	19298	19771	14936	15293	42814	36476	43907	39476
Contigs (>=0 bp)	28856	55524	38453	24038	63381	89598	130457	64158
Contigs (>=1000 bp)	9415	6336	7241	7717	19949	12135	17705	17125
Contigs (>=5000 bp)	1765	885	1692	1777	4054	1863	3146	3552
Contigs (>=10000 bp)	908	430	1012	1006	2263	1089	1901	2111
Contigs (>=25000 bp)	297	111	435	352	889	440	969	926
Contigs (>=50000 bp)	107	32	160	120	355	146	469	394
Largest contigs	289655	311703	708222	505697	321781	285207	591622	657432
Total length	54961438	32637685	56249322	54665552	129575865	72142446	133452776	126551027

trimming. The processed reads obtained from the previous step were used as the input for part 2 (Basic Protocol-2), with the results of this step showcased in Table 2, Figure 3, and Figure 4. Detailed explanations of the various MetaQUAST parameters are available elsewhere.[25] Among these parameters, N50 plays a pivotal role in assessing the accuracy of assembly files generated by the four assembler packages. Part 3 (Basic Protocol-3) yields results presented in Tables 3 and 4, focusing on the completeness and contamination percentages, which are key factors in evaluating the quality of Metagenome-Assembled Genomes (MAGs). Table 3 offers insights into the completeness and contamination percentages of MAGs, both during the initial phases and after refinement. To delve deeper into MAG quality assessment, we recommend consulting the article authored by Bowers

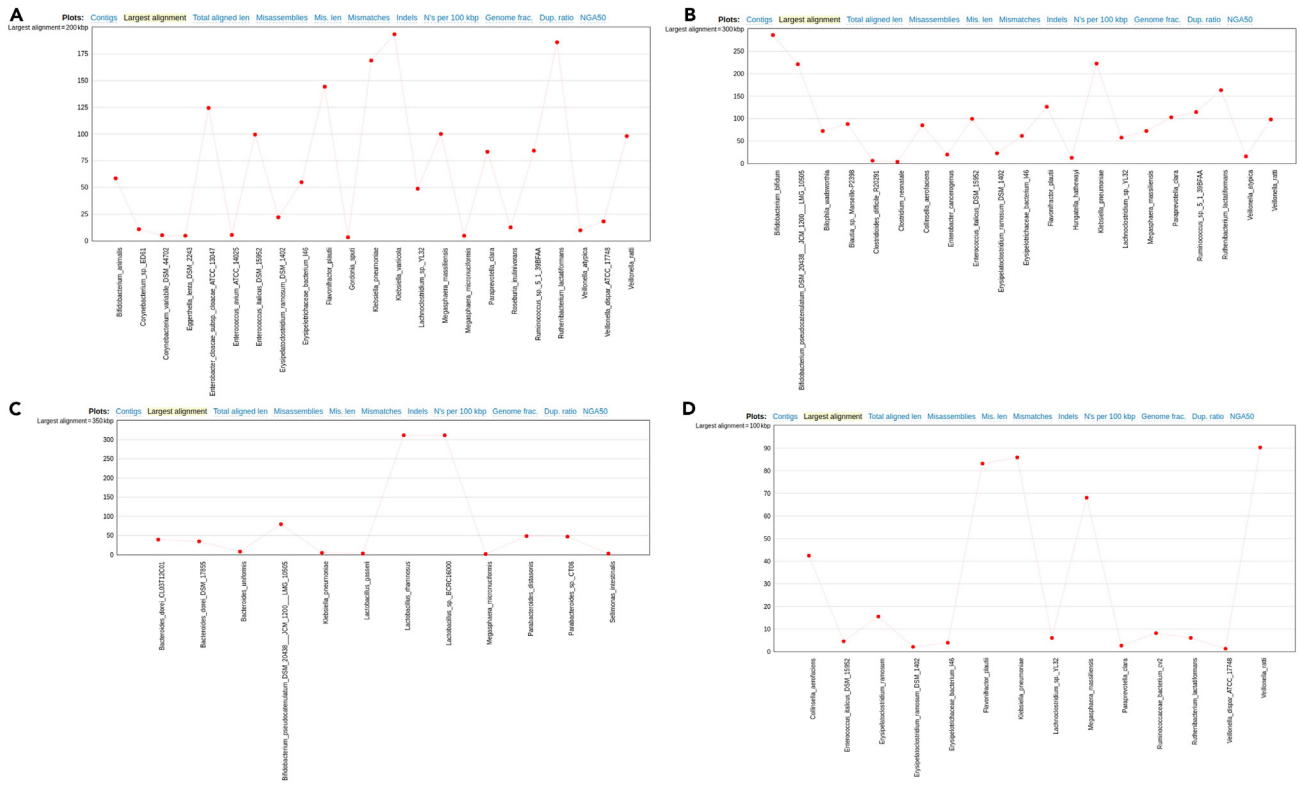


Figure 3. Metaquast statistics of assembly file generated from sample SRR23604271
(A) Idba_ud assembler, (B) megahit assembler, (C) metaspades, and (D) metavelvet.

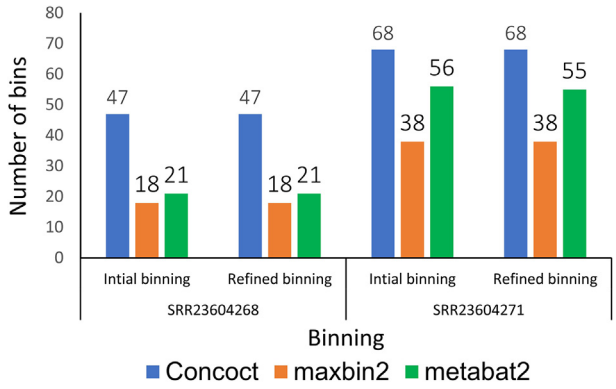


Figure 4. The number of initial bins and refined bins generated by three binning packages: concoct, maxbin2, and metabit2

et al.[35] This section also produces several informative figures (Figures 5, 6, 7, and 8) at various stages of MAG construction, refinement, and reassembly, with detailed descriptions available in publications elsewhere.[26,27] The results of part 4 (Basic Protocol-4) are presented in Tables 5, 6, and 7, encompassing the annotation (Table 5) and abundance (Table 6) of MAGs, showcasing the number of genes and the abundance of each MAG in the respective sample.

Furthermore, Table 7 provides insights into antibiotic resistance gene (ARG) profiling, offering information about the antibiotic resistance genes and associated antibiotics carried by each MAG. In summary, this comprehensive protocol guides users through a detailed exploration of each MAG, including taxonomic assignment, relative abundance, gene count, and ARG profiling. It is a valuable resource for researchers seeking a comprehensive understanding of their metagenomic data.

Table 3. Completeness and contamination of MAGs after reassembled using metaWRAP and CheckM

Original bins							Reassembled bins				
Sample ID	Bins	Completeness	Contamination	GC	Lineage	N50	Completeness	Contamination	GC	Lineage	N50
SRR23604268	bin.1	91.05	2.112	0.598	Bifidobacteriaceae	33898	91.68	1.075	0.597	Bifidobacteriaceae	49727
	bin.2	97.55	0.322	0.578	Enterobacteriaceae	20283	98.18	0.037	0.578	Enterobacteriaceae	58920
	bin.3	94.56	1.135	0.472	Bacteroidales	18938	94.56	1.135	0.472	Bacteroidales	18938
	bin.4	97.01	0.719	0.563	Bifidobacteriaceae	57499	97.01	0.719	0.563	Bifidobacteriaceae	57499
	bin.5	98.22	0.232	0.346	Lactobacillus	31587	98.22	0.155	0.346	Lactobacillus	32110
	bin.6	99.42	0.15	0.45	Bacteroidales	153055	99.42	0.15	0.45	Bacteroidales	153055
	bin.7	99.45	0	0.466	Lactobacillus	297865	99.45	0	0.466	Lactobacillus	297865
	bin.8	95.89	3.283	0.587	Bifidobacteriaceae	68075	95.94	3.264	0.587	Bifidobacteriaceae	95259
	bin.9	98.83	0.877	0.456	Lachnospiraceae	58776	98.83	0.877	0.456	Lachnospiraceae	58776
SRR23604271	bin.1	99.88	0	0.426	Selenomonadales	43613	99.88	0	0.426	Selenomonadales	43613
	bin.2	98.22	0	0.601	Deltaproteobacteria	20538	98.22	0	0.601	Deltaproteobacteria	20538
	bin.3	98.24	2.144	0.441	Lachnospiraceae	63166	98.24	2.144	0.441	Lachnospiraceae	75461
	bin.4	100	0.806	0.6	Actinobacteria	93588	100	0.806	0.6	Actinobacteria	93588
	bin.5	96.59	0.157	0.565	Clostridiales	54297	96.59	0.157	0.565	Clostridiales	54297
	bin.6	98.88	0	0.482	Bacteroidales	66456	98.88	0	0.482	Bacteroidales	66456
	bin.7	100	0	0.51	Selenomonadales	82507	100	0	0.51	Selenomonadales	82507
	bin.8	99.53	0	0.627	Bifidobacteriaceae	281888	99.53	0	0.627	Bifidobacteriaceae	281888
	bin.9	98.65	0.134	0.617	Clostridiales	95143	98.65	0.134	0.617	Clostridiales	95143
	bin.10	97.46	4.035	0.491	Clostridiales	72857	97.74	3.954	0.49	Clostridiales	78144
	bin.11	98.11	1.886	0.442	Bacteria	28911	98.11	1.886	0.442	Bacteria	28911
	bin.12	99.31	0.227	0.564	Bifidobacteriaceae	138784	99.31	0.227	0.564	Bifidobacteriaceae	138784
	bin.13	94.89	0.632	0.506	Clostridiales	50204	95.23	0.69	0.506	Clostridiales	56306
	bin.14	98.3	0.22	0.396	Bacilli	48006	98.34	0.044	0.396	Bacilli	48050
	bin.15	98.13	0.621	0.626	Proteobacteria	59267	98.13	0.621	0.626	Proteobacteria	59267
	bin.16	97.27	1.169	0.43	Lachnospiraceae	83330	97.36	0.584	0.43	Lachnospiraceae	93496
	bin.17	94.61	0.74	0.605	Bifidobacteriaceae	15679	94.61	0.74	0.605	Bifidobacteriaceae	15679

Table 4. Taxonomic assignment and associated lineage information of MAGs determined using GTDB-Tk database

Bins	Classification	fastani_reference	fastani_taxonomy	fastani_ani	fastani_af
SRR23604268_bin.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium longum	GCF_000196555.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium longum	98.34	0.75
SRR23604268_bin.2	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Enterobacterales; f__Enterobacteriaceae; g__Klebsiella;s__Klebsiella pneumoniae	GCF_000742135.1	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Enterobacterales; f__Enterobacteriaceae; g__Klebsiella; s__Klebsiella pneumoniae	99.1	0.92
SRR23604268_bin.3	d__Bacteria;p__Bacteroidota; c__Bacteroidia;o__Bacteroidales; f__Bacteroidaceae;g__Bacteroides; s__Bacteroides uniformis	GCF_000154205.1	d__Bacteria; p__Bacteroidota; c__Bacteroidia; o__Bacteroidales; f__Bacteroidaceae; g__Bacteroides; s__Bacteroides uniformis	98.22	0.89
SRR23604268_bin.4	d__Bacteria;p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium pseudocatenulatum	GCF_001025215.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium pseudocatenulatum	97.76	0.79
SRR23604268_bin.5	d__Bacteria;p__Firmicutes; c__Bacilli;o__Lactobacillales; f__Lactobacillaceae; g__Lactobacillus; s__Lactobacillus paragasseri	GCF_003584685.1	d__Bacteria; p__Firmicutes; c__Bacilli;o__Lactobacillales; f__Lactobacillaceae; g__Lactobacillus; s__Lactobacillus paragasseri	98.38	0.92
SRR23604268_bin.6	d__Bacteria;p__Bacteroidota; c__Bacteroidia;o__Bacteroidales; f__Tannerellaceae; g__Parabacteroides; s__Parabacteroides distasonis	GCF_000012845.1	d__Bacteria; p__Bacteroidota; c__Bacteroidia; o__Bacteroidales; f__Tannerellaceae; g__Parabacteroides; s__Parabacteroides distasonis	97.42	0.81
SRR23604268_bin.7	d__Bacteria;p__Firmicutes; c__Bacilli; o__Lactobacillales; f__Lactobacillaceae; g__Lacticaseibacillus; s__Lacticaseibacillus rhamnosus	GCF_900636965.1	d__Bacteria; p__Firmicutes; c__Bacilli; o__Lactobacillales; f__Lactobacillaceae; g__Lacticaseibacillus; s__Lacticaseibacillus rhamnosus	99.73	0.98
SRR23604268_bin.8	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium breve	GCF_001025175.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium breve	98.7	0.87
SRR23604268_bin.9	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Sellimonas; s__Sellimonas intestinalis	GCF_001280875.1	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Sellimonas; s__Sellimonas intestinalis	99.67	0.97

(*Continued on next page*)

Table 4. *Continued*

Bins	Classification	fastani_reference	fastani_taxonomy	fastani_ani	fastani_af
SRR23604271_bin.1	d__Bacteria; p__Firmicutes_C; c__Negativicutes; o__Veillonellales; f__Veillonellaceae; g__Veillonella_A; s__Veillonella_A sp000431435	GCA_008679445.1	d__Bacteria; p__Firmicutes_C; c__Negativicutes; o__Veillonellales; f__Veillonellaceae; g__Veillonella_A; s__Veillonella_A sp000431435	99.32	0.86
SRR23604271_bin.10	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Enterocloster; s__Enterocloster bolteae	GCF_002234575.2	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Enterocloster; s__Enterocloster bolteae	99.03	0.83
SRR23604271_bin.11	d__Bacteria; p__Firmicutes; c__Bacilli; o__Erysipelotrichales; f__Erysipelotrichaceae; g__Clostridium_AQ; s__Clostridium_AQ innocuum	GCF_012317185.1	d__Bacteria; p__Firmicutes; c__Bacilli; o__Erysipelotrichales; f__Erysipelotrichaceae; g__Clostridium_AQ; s__Clostridium_AQ innocuum	99.92	0.98
SRR23604271_bin.12	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium pseudocatenulatum	GCF_001025215.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium pseudocatenulatum	98.16	0.88
SRR23604271_bin.13	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Enterocloster; s__Enterocloster aldenensis	GCF_003434055.1	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Enterocloster; s__Enterocloster aldenensis	99.5	0.95
SRR23604271_bin.14	d__Bacteria; p__Firmicutes; c__Bacilli; o__Lactobacillales; f__Enterococcaceae; g__Enterococcus_G; s__Enterococcus_G italicus	GCF_000185365.1	d__Bacteria; p__Firmicutes; c__Bacilli; o__Lactobacillales; f__Enterococcaceae; g__Enterococcus_G; s__Enterococcus_G italicus	98.27	0.87
SRR23604271_bin.15	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Burkholderiales; f__Burkholderiaceae; g__Sutterella; s__Sutterella wadsworthensis_A	GCF_000297775.1	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Burkholderiales; f__Burkholderiaceae; g__Sutterella; s__Sutterella wadsworthensis_A	98.82	0.93
SRR23604271_bin.16	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Ruminococcus_B; s__Ruminococcus_B gnavus	GCF_008121495.1	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Ruminococcus_B; s__Ruminococcus_B gnavus	97.19	0.81
SRR23604271_bin.17	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium animalis	GCF_000260715.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium animalis	95.67	0.92

(*Continued on next page*)

Table 4. *Continued*

Bins	Classification	fastani_reference	fastani_taxonomy	fastani_ani	fastani_af
SRR23604271_bin.18	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Blautia_A; s__Blautia_A wexlerae	GCF_000484655.1	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Blautia_A; s__Blautia_A wexlerae	97.9	0.77
SRR23604271_bin.2	d__Bacteria; p__Desulfobacterota_I; c__Desulfovibrionia; o__Desulfovibrionales; f__Desulfovibrionaceae; g__Bilophila;s__Bilophila wadsworthia	GCF_000701705.1	d__Bacteria; p__Desulfobacterota_I; c__Desulfovibrionia; o__Desulfovibrionales; f__Desulfovibrionaceae; g__Bilophila; s__Bilophila wadsworthia	98.89	0.9
SRR23604271_bin.3	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Lachnospirales; f__Lachnospiraceae; g__Sellimonas;s_	N/A	N/A	N/A	N/A
SRR23604271_bin.4	d__Bacteria; p__Actinobacteriota; c__Coriobacteriia; o__Coriobacteriales; f__Coriobacteriaceae; g__Collinsella;s__	N/A	N/A	N/A	N/A
SRR23604271_bin.5	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Oscillospirales; f__Ruminococcaceae; g__Ruthenibacterium; s__Ruthenibacterium lactatiformans	GCF_000949455.1	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Oscillospirales; f__Ruminococcaceae; g__Ruthenibacterium; s__Ruthenibacterium lactatiformans	98.81	0.71
SRR23604271_bin.6	d__Bacteria; p__Bacteroidota; c__Bacteroidia; o__Bacteroidales; f__Bacteroidaceae; g__Paraprevotella; s__Paraprevotella clara	GCF_000233955.1	d__Bacteria; p__Bacteroidota; c__Bacteroidia; o__Bacteroidales; f__Bacteroidaceae; g__Paraprevotella; s__Paraprevotella clara	97.36	0.83
SRR23604271_bin.7	d__Bacteria; p__Firmicutes_C; c__Negativicutes; o__Veillonellales; f__Megasphaeraceae; g__Megasphaera; s__Megasphaera sp001546855	GCF_001546855.1	d__Bacteria; p__Firmicutes_C; c__Negativicutes; o__Veillonellales; f__Megasphaeraceae; g__Megasphaera; s__Megasphaera sp001546855	99.95	0.98
SRR23604271_bin.8	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium bifidum	GCF_001025135.1	d__Bacteria; p__Actinobacteriota; c__Actinomycetia; o__Actinomycetales; f__Bifidobacteriaceae; g__Bifidobacterium; s__Bifidobacterium bifidum	98.97	0.94
SRR23604271_bin.9	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Oscillospirales; f__Oscillospiraceae; g__Flavonifractor; s__Flavonifractor plautii	GCF_000239295.1	d__Bacteria; p__Firmicutes_A; c__Clostridia; o__Oscillospirales; f__Oscillospiraceae; g__Flavonifractor; s__Flavonifractor plautii	98.52	0.83

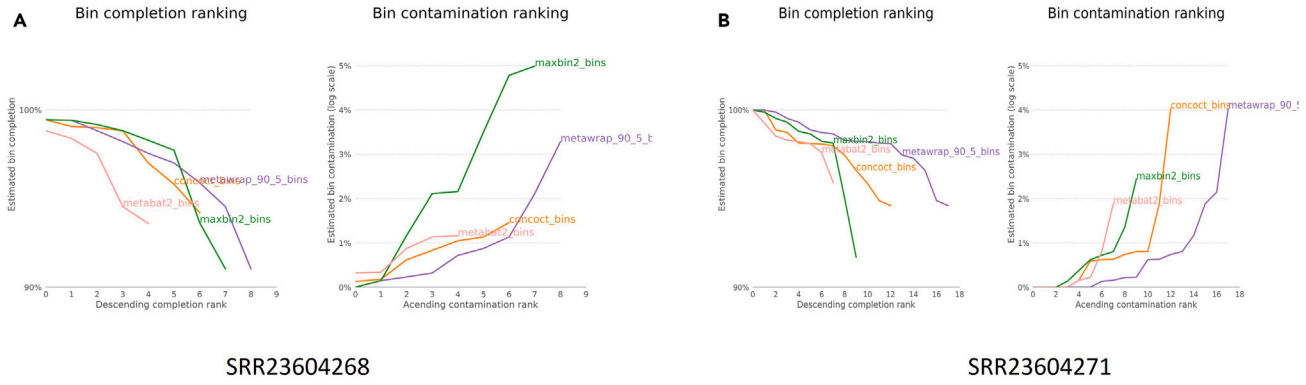


Figure 5. Bin refinement graph showing completion and contamination ranking

(A) and (B) represent sample SRR23604268 and SRR23604271, respectively.

LIMITATIONS

MAG construction is indeed a multistep process, and it is subject to various limitations, including.

Sequence depth and coverage heterogenicity

We have used deep shotgun sequences (> 20 million reads) for MAG construction here. However, shallow sequence reads (< 10 million) may lead to incomplete or low-quality MAGs with low completeness and high contamination.

Metagenomic assembly

In this tutorial, we have used four assemblers to construct assembly contig from metagenomic reads. Other metagenomic assemblers/ algorithms might generate better results.

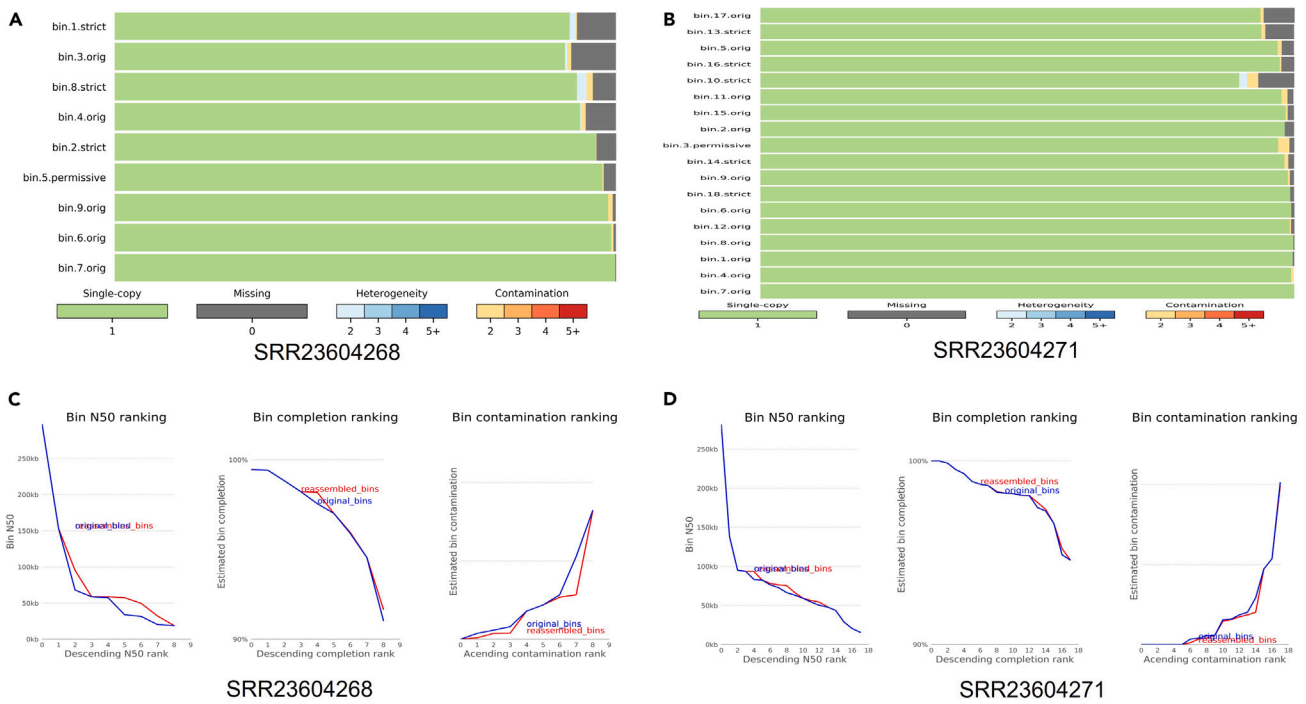


Figure 6. Showing graphs obtained after reassembled binning step

(A) and (B) represent the CheckM output graph of samples SRR23604268 and SRR23604271, respectively.

(C) and (D) indicate different bin ranking parameters of samples SRR23604268 and SRR23604271, respectively.

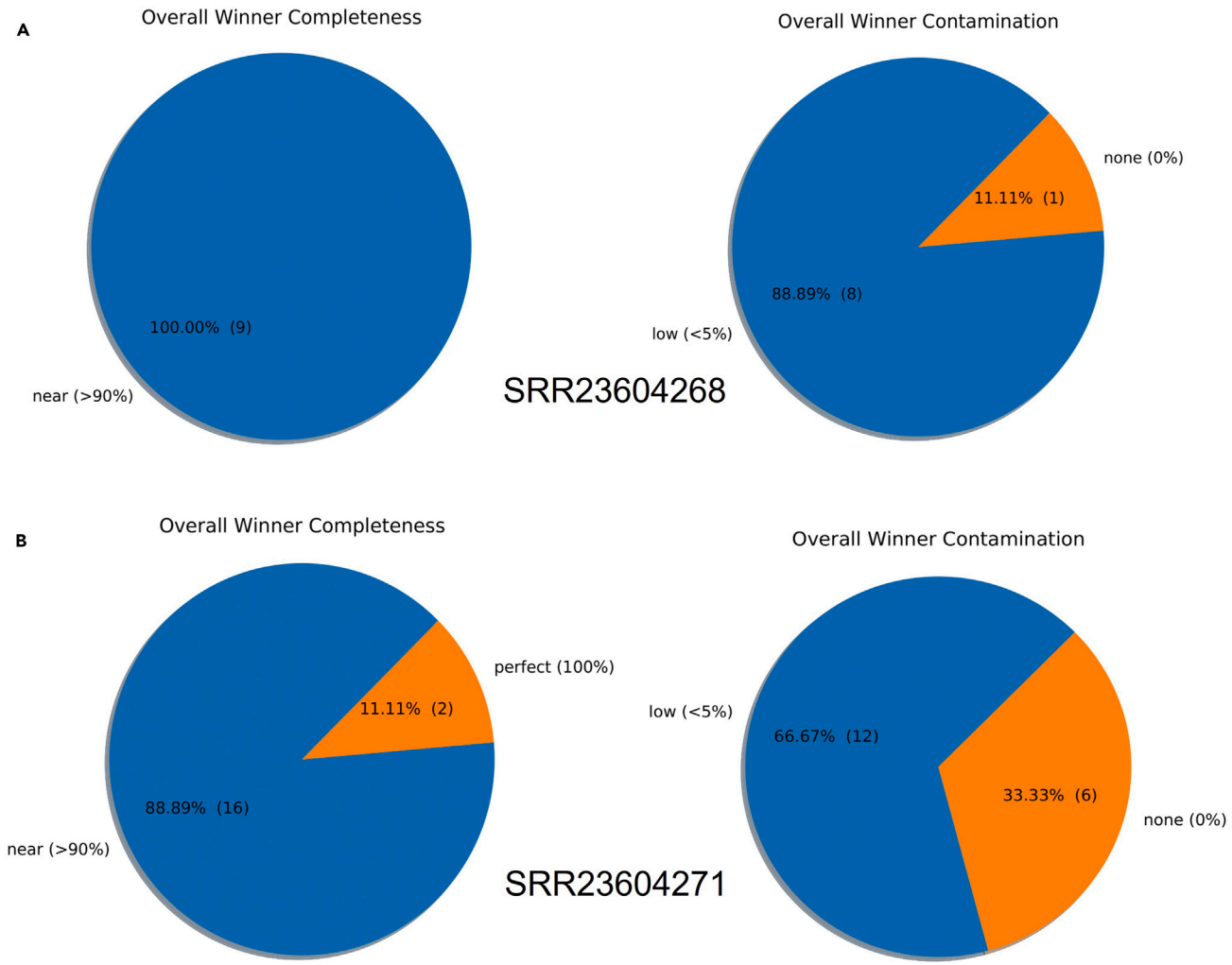


Figure 7. De-replicated bins showing overall completeness and contamination
(A) and (B) indicate samples SRR23604268 and SRR23604271, respectively.

Software and database version

Changes in software or database versions may impact the results of MAG construction. It is crucial to stay updated with the latest versions and adjustments in the coding to incorporate these changes effectively.

Gene annotation

Here, we used Prokka for gene annotation from MAG. Exploring alternative annotation pipelines may offer additional insights or variations in gene annotations.

ARG detection

Changes or updates in antibiotic resistance gene (ARG) databases can affect the detection and interpretation of ARGs in MAGs. Regular updates and validation against current databases are necessary to ensure accurate ARG detection.

TROUBLESHOOTING

We have presented a straightforward and robust workflow for generating MAGs and their associated functional profiles, spanning from raw sequence data to the final results.

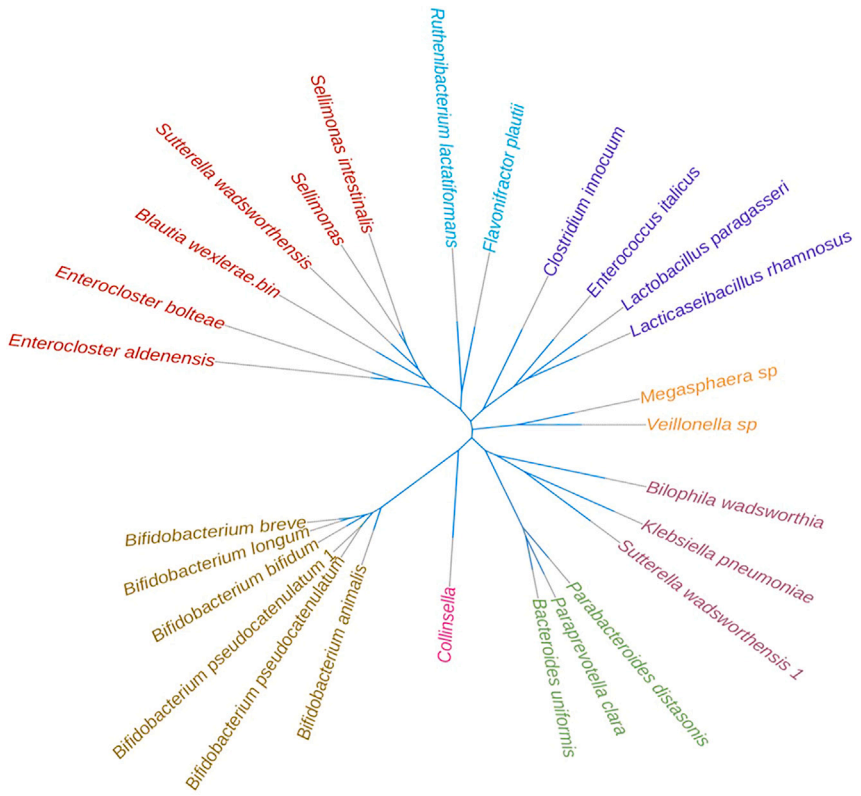


Figure 8. Showing the phylogenetic tree of all MAGs generated by phylophlan package

Table 5. Annotation information of constructed MAGs using prokka

Constructed MAGs	CDS	Gene	misc_RNA	rRNA	Repeat region	tRNA	tmRNA
Bacteroides_uniformis	3137	3178	15	1	1	25	0
Bifidobacterium_animalis	1539	1579	9	0	2	30	1
Bifidobacterium_bifidum	1737	1795	8	0	0	49	1
Bifidobacterium_breve	1894	1961	15	0	0	51	1
Bifidobacterium_longum	2151	2217	14	1	0	50	1
Bifidobacterium_pseudocatenulatum	1888	1952	11	1	1	51	1
Bifidobacterium_pseudocatenulatum_1	2155	2208	12	1	4	39	1
Bilophila_wadsworthia	3545	3616	18	0	5	52	1
Blautia_wexlerae	3773	3879	64	0	0	41	1
Clostridium_innocuum	3691	3781	43	0	3	46	1
Collinsella_sp.	1798	1866	9	0	0	57	2
Enterocloster_aldenensis	4766	4834	36	0	2	32	0
Enterocloster_bolteae	5584	5691	62	0	3	45	0
Enterococcus_italicus	2064	2124	37	0	1	22	1
Flavonifractor_plautii	3544	3629	46	0	1	39	0
Klebsiella_pneumoniae	4703	4872	109	1	0	58	1
Lacticaseibacillus_rhamnosus	2718	2809	31	4	0	55	1
Lactobacillus_paragasseri	1854	1917	33	0	1	29	1
Megasphaera_sp.	2224	2311	35	1	11	50	1
Parabacteroides_distasonis	4027	4177	83	6	1	60	1
Paraprevotella_clara	3411	3474	14	0	0	48	1
Ruthenibacterium_lactatiformans	3812	3895	40	0	1	42	1
Sellimonas_sp.	2802	2870	33	0	0	34	1
Sellimonas_intestinalis	2735	2807	30	0	0	41	1
Ruminococcus_gnavus	2726	2787	28	0	1	32	1
Sutterella_wadsworthensis	1997	2081	23	0	0	60	1
Veillonella_sp.	2545	2648	43	8	1	51	1

Table 6. Abundance of MAGs in respective sample assessed my mapping-based method

Samples	MAGs	Abundance
SRR23604268	*Bacteroides_uniformis*	2.30%
	Bifidobacterium_breve	38.61%
	Bifidobacterium_longum	38.38%
	Bifidobacterium_pseudocatenulatum	13.51%
	Klebsiella_pneumoniae	2.27%
	Lacticaseibacillus_rhamnosus	4.09%
	Lactobacillus_paragasseri	0.53%
	Parabacteroides_distasonis	4.19%
	Sellimonas_intestinalis	0.96%
SRR23604271	*Bifidobacterium_animalis*	4.70%
	Bifidobacterium_bifidum	17.33%
	Bifidobacterium_pseudocatenulatum_1	11.92%
	Bilophila_wadsworthia	0.84%
	Blautia_wexlerae	22.08%
	Clostridium_innocuum	0.73%
	Collinsella	3.07%
	Enterocloster_aldenensis	1.41%
	Enterocloster_bolteae	1.78%
	Enterococcus_italicus	0.38%
	Flavonifractor_plautii	3.65%
	Megasphaera_sp	9.15%
	Paraprevotella_clara	0.62%
	Ruminococcus_gnavus	1.78%
	Ruthenibacterium_lactatiformans	1.16%
	Sellimonas	2.38%
	Sutterella_wadsworthensis	2.32%
	Veillonella_sp	1.87%

Problem 1

Working in a Conda environment is user-friendly; however, installing multiple packages with a single command line may sometimes pose challenges.

Potential solution

In such a situation, users should go for the option to install individual packages one by one in the same environment using the command 'conda install -c bioconda packagename==version'.

Problem 2

The channel priority in Conda is crucial and may interrupt smooth package execution.

Potential solution

To ensure the correct priority order, consider adding 'bioconda' before 'conda-forge' in your channel configuration step (this can be verified using the command 'conda config –show channels').

Problem 3

The installation of QUAST in a conda environment sometimes creates issues.

Potential solution

In that case, users can install it through Python using command- 'pip install quast', but make sure that you are in the conda environment.

Table 7. Antibiotic resistance profile showing both susceptible and resistant MAGs

MAGs	Genotype	Predicted phenotype	Antibiotics
Bacteroides_uniformis	None	Susceptible	-
Bifidobacterium_animalis	None	Susceptible	-
Bifidobacterium_bifidum	None	Susceptible	-
Bifidobacterium_breve	None	Susceptible	-
Bifidobacterium_longum	None	Susceptible	-
Bifidobacterium_pseudocatenulatum	None	Susceptible	-
Bifidobacterium_pseudocatenulatum_1	None	Susceptible	-
Bilophila_wadsworthia	None	Susceptible	-
Blautia_wexlerae.bin	None	Susceptible	-
Clostridium_innocuum	None	Susceptible	-
Collinsella	None	Susceptible	-
Enterocloster_aldenensis	None	Susceptible	-
Enterocloster_bolteae	erm(B), erm(B)	unknown[erm(B)_12_U18931], erythromycin, azithromycin	Erythromycin, Lincomycin, Clindamycin, Quinupristin, Pristinamycin IA, Virginiamycin S
Enterococcus_italicus	None	Susceptible	-
Flavonifractor_plautii	None	Susceptible	-
Klebsiella_pneumoniae	blaSHV-89, blaSHV-89, OqxA, OqxA, OqxB, OqxB	unknown[blaSHV-89_1_DQ193536], ampicillin, unknown [OqxA_1_EU370913], unknown [OqxB_1_EU370913]	Amoxicillin, Ampicillin, Cephalothin, Piperacillin, Ticarcillin, Chloramphenicol, Nalidixic acid, Ciprofloxacin, Trimethoprim
Lacticaseibacillus_rhamnosus	None	Susceptible	-
Lactobacillus_paragasseri	None	Susceptible	-
Megasphaera_sp	blaACI-1, blaACI-1, tet(W), tet(W)	unknown[blaACI-1_1_AJ007350], ampicillin, tetracycline, unknown[tet(W)_5_AJ427422]	Amoxicillin, Ampicillin, Ticarcillin, Ceftazidime, Cefotaxime, Doxycycline, Tetracycline, Minocycline
Parabacteroides_distasonis	None	Susceptible	-
Paraprevotella_clara	tet(Q), tet(Q)	unknown[tet(Q)_1_L33696], tetracycline	Doxycycline, Tetracycline, Minocycline
Ruthenibacterium_lactatiformans	None	Susceptible	-
Sellimonas	None	Susceptible	-
Sellimonas_intestinalis	tet(M), tet(M)	unknown[tet(M)_8_X04388], tetracycline	Doxycycline, Tetracycline, Minocycline
Sutterella_wadsworthensis	lnu(C), lnu(C)	lincomycin, unknown [lnu(C)_1_AY928180]	Lincomycin
Sutterella_wadsworthensis_1	None	Susceptible	-
Veillonella_sp	tet(M), tet(M), tet(O), tet(O)	tetracycline, unknown [tet(M)_1_X92947], unknown[tet(O)_1_M18896]	Doxycycline, Tetracycline, Minocycline

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources must be directed to lead contact, Pratik Banerjee (pratik@illinois.edu).

Technical contact

For technical information, including software and code, contact Goutam Banerjee (goutamb@illinois.edu).

Materials availability

This is a bioinformatics pipeline and thus we have not used any biological material or reagents.

Data and code availability

Here we have used public data from NCBI [SRA data sets: SRR23604268 and SRR23604271) under the bioproject number PRJNA938144]. The codes used in this tutorial are given in each section (part 1 to part 4).

ACKNOWLEDGMENTS

This work was partially supported by a USDA-NIFA project (project # ILLU-698-981).

AUTHOR CONTRIBUTIONS

Conceptualization, G.B. and P.B.; data curation, G.B. and S.R.P.; data analysis, G.B.; investigation, G.B.; methodology, G.B. and P.B.; software, G.B.; validation, G.B. and P.B.; writing original draft, G.B., P.B., and S.R.P.; table preparation, S.R.P.; manuscript review and editing, G.B. and P.B.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Alsayed, A.R., Abed, A., Khader, H.A., Al-Shdifat, L.M.H., Hasoun, L., Al-Rshaidat, M.M.D., Alkhatib, M., and Zihlif, M. (2023). Molecular Accounting and Profiling of Human Respiratory Microbial Communities: Toward Precision Medicine by Targeting the Respiratory Microbiome for Disease Diagnosis and Treatment. Int. J. Mol. Sci. *24*, 4086. https://doi.org/10.3390/ijms24044086.

2. Banerjee, S., and van der Heijden, M.G.A. (2023). Soil microbiomes and one health. Nat. Rev. Microbiol. *21*, 6–20. https://doi.org/10.1038/s41579-022-00779-w.

3. Galand, P.E., Ruscheweyh, H.-J., Salazar, G., Hochart, C., Henry, N., Hume, B.C.C., Oliveira, P.H., Perdereau, A., Labadie, K., Belser, C., et al. (2023). Diversity of the Pacific Ocean coral reef microbiome. Nat. Commun. *14*, 3039. https://doi.org/10.1038/s41467-023-38500-x.

4. Chen, P., Sun, W., and He, Y. (2020). Comparison of the next-generation sequencing (NGS) technology with culture methods in the diagnosis of bacterial and fungal infections. J. Thorac. Dis. *12*, 4924–4929. https://doi.org/10.21037/jtd-20-930.

5. Banerjee, G., Agarwal, S., Marshall, A., Jones, D.H., Sulaiman, I.M., Sur, S., and Banerjee, P. (2022). Application of advanced genomic tools in food safety rapid diagnostics: challenges and opportunities. Curr. Opin. Food Sci. *47*, 100886. https://doi.org/10.1016/j.cofs.2022.100886.

6. Reji, L., and Francis, C.A. (2020). Metagenome-assembled genomes reveal unique metabolic adaptations of a basal marine Thaumarchaeota lineage. ISME J. *14*, 2105–2115. https://doi.org/10.1038/s41396-020-0675-6.

7. Nascimento Lemos, L., Manoharan, L., William Mendes, L., Monteiro Venturini, A., Satler Pylro, V., and Tsai, S.M. (2020). Metagenome assembled-genomes reveal similar functional profiles of CPR/Patescibacteria phyla in soils. Environ. Microbiol. Rep. *12*, 651–655. https://doi.org/10.1111/1758-2229.12880.

8. Li, C., Li, X., Guo, R., Ni, W., Liu, K., Liu, Z., Dai, J., Xu, Y., Abduriyim, S., Wu, Z., et al. (2023). Expanded catalogue of metagenome-assembled genomes reveals resistome characteristics and athletic performance-associated microbes in horse. Microbiome *11*, 7. https://doi.org/10.1186/s40168-022-01448-z.

9. Lv, Q.-B., Li, S., Zhang, Y., Guo, R., Wang, Y.-C., Peng, Y., and Zhang, X.-X. (2022). A thousand metagenome-assembled genomes of Akkermansia reveal phylogroups and geographical and functional variations in the human gut. Front. Cell. Infect. Microbiol. *12*, 957439. https://doi.org/10.3389/fcimb.2022.957439.

10. Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., Richardson, P.M., Solovyev, V.V., Rubin, E.M., Rokhsar, D.S., and Banfield, J.F. (2004). Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature *428*, 37–43. https://doi.org/10.1038/nature02340.

11. Quince, C., Walker, A.W., Simpson, J.T., Loman, N.J., and Segata, N. (2017). Shotgun metagenomics, from sampling to analysis. Nat. Biotechnol. *35*, 833–844. https://doi.org/10.1038/nbt.3935.

12. Balaji, A., Sapoval, N., Seto, C., Leo Elworth, R.A., Fu, Y., Nute, M.G., Savidge, T., Segarra, S., and Treangen, T.J. (2022). KOMB: K-core based de novo characterization of copy number variation in microbiomes. Comput. Struct. Biotechnol. J. *20*, 3208–3222. https://doi.org/10.1016/j.csbj.2022.06.019.

13. Martin, S., Ayling, M., Patrono, L., Caccamo, M., Murcia, P., and Leggett, R.M. (2023). Capturing variation in metagenomic assembly graphs with MetaCortex. Bioinformatics *39*, btad020. https://doi.org/10.1093/bioinformatics/btad020.

14. Kroeger, M.E., Delmont, T.O., Eren, A.M., Meyer, K.M., Guo, J., Khan, K., Rodrigues, J.L.M., Bohannan, B.J.M., Tringe, S.G., Borges, C.D., et al. (2018). New biological insights into how deforestation in Amazonia affects soil microbial communities using metagenomics and metagenome-assembled genomes. Front. Microbiol. *9*, 1635. https://doi.org/10.3389/fmicb.2018.01635.

15. Wang, J., Qi, J., Zhao, H., He, S., Zhang, Y., Wei, S., and Zhao, F. (2013). Metagenomic sequencing reveals microbiota and its functional potential associated with periodontal disease. Sci. Rep. *3*, 1843. https://doi.org/10.1038/srep01843.

16. Chen, Z., Liu, W.-S., Zhong, X., Zheng, M., Fei, Y.-h., He, H., Ding, K., Chao, Y., Tang, Y.-T., Wang, S., and Qiu, R. (2021). Genome-and community-level interaction insights into the ecological role of archaea in rare earth element mine drainage in South China. Water Res. *201*, 117331. https://doi.org/10.1016/j.watres.2021.117331.

17. Nurk, S., Meleshko, D., Korobeynikov, A., and Pevzner, P.A. (2017). metaSPAdes: a new versatile metagenomic assembler. Genome Res. *27*, 824–834. https://doi.org/10.1101/gr.213959.116.

18. Li, D., Liu, C.-M., Luo, R., Sadakane, K., and Lam, T.-W. (2015). MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics *31*, 1674–1676. https://doi.org/10.1093/bioinformatics/btv033.

19. Peng, Y., Leung, H.C.M., Yiu, S.-M., and Chin, F.Y.L. (2012). IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics *28*, 1420–1428. https://doi.org/10.1093/bioinformatics/bts174.

20. Zhang, A., de Ángel Solá, D., Acevedo Flores, M., Cao, L., Wang, L., Kim, J.G., Tarr, P.I., Warner, B.B., Rosario Matos, N., and Wang, L. (2023). Infants exposed in utero to Hurricane Maria have gut microbiomes with reduced diversity and altered metabolic capacity. mSphere *8*, e0013423. https://doi.org/10.1128/msphere.00134-23.

21. Chen, S., Zhou, Y., Chen, Y., and Gu, J. (2018). fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics *34*, i884–i890. https://doi.org/10.1093/bioinformatics/bty560.

22. S, A. (2020). FastQC: A Quality Control Tool for High Throughput Sequence Data. http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

23. Langmead, B., and Salzberg, S.L. (2012). Fast gapped-read alignment with Bowtie 2. Nat. Methods *9*, 357–359. https://doi.org/10.1038/nmeth.1923.

24. Namiki, T., Hachiya, T., Tanaka, H., and Sakakibara, Y. (2012). MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic Acids Res. *40*, e155. https://doi.org/10.1093/nar/gks678.

25. Mikheenko, A., Saveliev, V., and Gurevich, A. (2016). MetaQUAST: evaluation of metagenome assemblies. Bioinformatics *32*, 1088–1090. https://doi.org/10.1093/bioinformatics/btv697.

26. Uritskiy, G.V., DiRuggiero, J., and Taylor, J. (2018). MetaWRAP—a flexible pipeline for genome-resolved metagenomic data analysis. Microbiome *6*, 158–170. https://doi.org/10.1186/s40168-018-0541-1.

27. Olm, M.R., Brown, C.T., Brooks, B., and Banfield, J.F. (2017). dRep: a tool for fast and accurate genomic comparisons that enables improved genome recovery from metagenomes through de-replication. ISME J. *11*, 2864–2868. https://doi.org/10.1038/ismej.2017.126.

28. Chaumeil, P.A., Mussig, A.J., Hugenholtz, P., and Parks, D.H. (2019). GTDB-Tk: a toolkit to classify genomes with the Genome Taxonomy Database. Bioinformatics *36*, 1925–1927. https://doi.org/10.1093/bioinformatics/btz848.

29. Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. Preprint at arXiv. https://doi.org/10.48550/arXiv.1303.3997.

30. Danecek, P., Bonfield, J.K., Liddle, J., Marshall, J., Ohan, V., Pollard, M.O., Whitwham, A., Keane, T., McCarthy, S.A., Davies, R.M., and Li, H. (2021). Twelve years of SAMtools and BCFtools. GigaScience *10*, giab008. https://doi.org/10.1093/gigascience/giab008.

31. Seemann, T. (2014). Prokka: rapid prokaryotic genome annotation. Bioinformatics *30*, 2068–2069. https://doi.org/10.1093/bioinformatics/btu153.

32. Asnicar, F., Thomas, A.M., Beghini, F., Mengoni, C., Manara, S., Manghi, P., Zhu, Q., Bolzan, M., Cumbo, F., May, U., et al. (2020). Precise phylogenetic analysis of microbial isolates and genomes from metagenomes using PhyloPhlAn 3.0. Nat. Commun. *11*, 2500. https://doi.org/10.1038/s41467-020-16366-7.

33. Bharat, A., Petkau, A., Avery, B.P., Chen, J.C., Folster, J.P., Carson, C.A., Kearney, A., Nadon, C., Mabon, P., Thiessen, J., et al. (2022). Correlation between phenotypic and in silico detection of antimicrobial resistance in Salmonella enterica in Canada using Staramr. Microorganisms *10*, 292. https://doi.org/10.3390/microorganisms10020292.

34. Parks, D.H., Chuvochina, M., Rinke, C., Mussig, A.J., Chaumeil, P.-A., and Hugenholtz, P. (2022). GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. Nucleic Acids Res. *50*, D785–D794. https://doi.org/10.1093/nar/gkab776.

35. Bowers, R.M., Kyrpides, N.C., Stepanauskas, R., Harmon-Smith, M., Doud, D., Reddy, T.B.K., Schulz, F., Jarett, J., Rivers, A.R., Eloe-Fadrosh, E.A., et al. (2017). Minimum information about a single amplified genome (MISAG) and a metagenome-assembled genome (MIMAG) of bacteria and archaea. Nat. Biotechnol. *35*, 725–731. https://doi.org/10.1038/nbt.3893.

36. Segata, N., Börnigen, D., Morgan, X.C., and Huttenhower, C. (2013). PhyloPhlAn is a new method for improved phylogenetic and taxonomic placement of microbes. Nat. Commun. *4*, 2304. https://doi.org/10.1038/ncomms3304.