Metagenomic Hi–C Protocols for Viral Genome Binning, Taxonomic Annotation, and Interaction Network Visualization

Yuxuan Du,[1,3,4] Yuqiu Wang,[2,3] and Fengzhu Sun[2,4]

[1]Department of Electrical Engineering, The University of Texas at San Antonio, San Antonio, Texas
[2]Department of Quantitative and Computational Biology, University of Southern California, Los Angeles, California
[3]These authors contributed equally to this work
[4]Corresponding authors: *yuxuan.du@utsa.edu*; *fsun@usc.edu*

Published in the Bioinformatics section

Metagenomic Hi-C (metaHi-C) links mobile genetic elements to their cellular hosts directly within complex microbial communities. Once shotgun and Hi-C libraries have been generated, however, the main challenges shift to the bioinformatics required for preprocessing, genome binning, taxonomic annotation, and network-level interpretation. Here, we present metaHi-C protocols that span from raw reads to downstream data analyses. Basic Protocol 1 describes quality control of shotgun and Hi-C reads, metagenomic assembly, Hi-C read mapping, and viral contig identification from assembled contigs. Basic Protocol 2 details the use of ViralCC to recover viral metagenome-assembled genomes (vMAGs) and infer virus–host linkages. Support Protocol 1 introduces NormCC and ImputeCC for normalization of raw Hi-C contacts and host genome binning. Support Protocols 2 and 3 describe taxonomic annotation of host MAGs with GTDB-Tk and viral bins with Virgo, respectively. Support Protocol 4 shows how to integrate these outputs in MetaHiCNet to generate cross-taxa and cross-bin Hi-C interaction networks. Together, these protocols provide a reproducible workflow for reconstructing viral and host genomes, assigning consistent taxonomies, and visualizing metaHi-C–derived virus–host interaction structure across diverse microbiomes. © 2026 The Author(s). *Current Protocols* published by Wiley Periodicals LLC.

Basic Protocol 1: Preprocessing raw metagenomic Hi-C data
Basic Protocol 2: Viral genome binning and virus–host interaction inference using ViralCC
Support Protocol 1: Host genome binning using ImputeCC
Support Protocol 2: Host MAG taxonomic annotation with GTDB–Tk
Support Protocol 3: Viral bin taxonomic annotation with Virgo
Support Protocol 4: Visualization of virus–host interaction networks with MetaHiCNet

Keywords: interaction network visualization • metagenome–assembled genomes • metagenomic Hi–C • viral contig binning • virus–host interactions

Current Protocols e70341, Volume 6
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpz1.70341

If you found this article helpful, please cite it.

How to cite this article:
Du, Y., Wang, Y., & Sun, F. (2026). Metagenomic Hi–C protocols for viral genome binning, taxonomic annotation, and interaction network visualization. *Current Protocols*, *6,* e70341.
doi: 10.1002/cpz1.70341

INTRODUCTION

Microbial communities underpin key processes in human health, agriculture, and the environment, yet the genetic reservoirs that shape their function are distributed across complex networks of hosts and mobile genetic elements, including bacteriophages and other viruses (Gilbert et al., 2014; Thompson et al., 2017). Short-read metagenomic sequencing has enabled assembly of metagenome-assembled genomes (MAGs) and viral MAGs (vMAGs), greatly expanding our view of uncultivated microbial and viral diversity (Du et al., 2023; Nayfach et al., 2021b; Parks et al., 2017). However, linking viral genomes to their cellular hosts in situ remains challenging. Traditional association methods, such as sequence similarity, CRISPR spacer matches, or co-abundance patterns, often suffer from low sensitivity, weak host resolution, or limited applicability to novel lineages (Edwards et al., 2016; Roux et al., 2016).

Metagenomic Hi-C (metaHi-C) overcomes many of these limitations by capturing physical DNA–DNA proximity in vivo (Du & Sun, 2023). Hi-C cross-links DNA fragments that co-reside in the same cell, so contacts between contigs can be used to infer host–virus relationships and to improve genome binning (Du et al., 2023; Du & Sun, 2023). However, metaHi-C analyses remain technically demanding. Raw Hi-C reads require specialized preprocessing; raw contact maps are subject to multiple biases; and downstream tools for binning, taxonomy, and visualization are often developed independently, with non-standard input formats (Du et al., 2023; Du & Sun, 2023; Sun et al., 2025). Consequently, many laboratories lack a unified, reproducible pipeline that takes raw reads through to virus–host networks that can be interpreted biologically.

This article presents a coherent set of computational protocols that address these gaps by chaining together widely used tools for preprocessing, binning, taxonomic annotation, and visualization of metaHi-C data. Basic Protocol 1 covers preprocessing of raw whole-genome shotgun (WGS) and Hi-C reads. Users begin with paired-end FASTQ files and perform read quality control, adapter and quality trimming, and duplicate removal using BBTools (Bushnell, 2014). Cleaned WGS reads are assembled with MEGAHIT or metaSPAdes to produce contigs (D. Li et al., 2015; Nurk et al., 2017), and Hi-C reads are mapped back to the assembly using BWA and samtools (H. Li, 2013; H. Li et al., 2009). Viral contigs are then identified using geNomad (Camargo et al., 2023), providing a set of candidate viral sequences for downstream analyses. Basic Protocol 2 introduces ViralCC, which integrates Hi-C–derived contact data with viral contig information to perform viral genome binning and virus–host interaction inference (Du et al., 2023). ViralCC's raw module builds contig-level Hi-C contact matrices and reports contig metadata. The bin module clusters viral contigs into draft viral bins (vMAGs) by jointly leveraging Hi-C contacts and a host proximity graph, while the link module infers virus–host linkages between viral and host contigs or bins (Du et al., 2023). Together, these steps transform assembled contigs and Hi-C mappings into structured outputs: vMAGs, host bins, and quantitative virus–host contact strengths.

To maximize the information in Hi-C data for host genome reconstruction, Support Protocol 1 describes how to normalize raw Hi-C contacts with NormCC and then perform host genome binning with ImputeCC (Du & Sun, 2023; Du et al., 2024). NormCC corrects key biases in raw contact counts and produces both a normalized contact matrix and contig-level information (Du & Sun, 2023). ImputeCC integrates this normalized Hi-C signal with single-copy marker gene information to generate high-quality host MAGs, complementing the viral bins produced by ViralCC (Du et al., 2024). Once viral and host MAGs are available, Support Protocol 2 and Support Protocol 3 provide standardized taxonomic annotations. Support Protocol 2 uses GTDB-Tk to assign GTDB taxonomy to host MAGs, enabling consistent placement of bacterial and archaeal genomes within a curated reference tree (Chaumeil et al., 2019). Support Protocol 3 describes the use of Virgo to annotate viral bins with ICTV-based taxonomic lineages (Riccardi et al., 2025). Together, these two protocols provide harmonized taxonomy for both sides of the virus–host interaction network.

Finally, Support Protocol 4 introduces MetaHiCNet, a web-based platform for visualizing normalized Hi-C interactions (Sun et al., 2025). Using contig, binning, taxonomy, and contact files exported from ViralCC, ImputeCC, GTDB-Tk, and Virgo, MetaHiCNet generates interactive cross-taxa interaction graphs and cross-bin networks (Sun et al., 2025). These visualizations facilitate exploration of overall community composition, identification of prominent virus–host pairs, and communication of results in a figure-ready format. In combination, the basic and support protocols guide readers from raw metaHi-C sequencing data to biologically interpretable virus–host interaction networks. The workflow is modular, each component can be adopted independently, but it is designed to be executed end-to-end, providing a reproducible framework for characterizing viral and host genomes, their taxonomies, and their interaction structure in diverse microbiomes.

For tutorial use, example files (paired WGS and Hi-C FASTQ files for Basic Protocol 1) and example intermediates for Basic Protocols and Support Protocols are available at: *https://doi.org/10.5281/zenodo.18461835*. All tutorial runs in this article were executed on a 64-bit Linux system with 64 GB RAM allocated. Commands assume input files are in the current working directory unless full paths are provided.

PREPROCESSING RAW METAGENOMIC HI-C DATA

BASIC PROTOCOL 1

This protocol describes the preprocessing workflow for metagenomic Hi-C data analysis, starting from raw sequencing reads. It assumes paired-end FASTQ inputs from both metagenomic whole-genome sequencing (WGS) and Hi-C experiments. The workflow performs (1) read quality control, (2) metagenomic assembly, (3) Hi-C read alignment, and (4) viral contig identification. Specifically, this protocol uses BBTools (Bushnell, 2014), MEGAHIT (D. Li et al., 2015), BWA (Li, 2013), and geNomad (Camargo et al., 2023) for these key preprocessing stages.

Necessary Resources

Hardware

A 64-bit Linux system with sufficient RAM to support the in-memory operations

The memory and disk consumption heavily depend on the input data, and in many cases cannot be estimated beforehand.

Software

Python
Java 7 or higher
BBTools

MEGAHIT
BWA
geNomad packages

Files

Paired-end FASTQ files for WGS and Hi-C sequencing

Input files can also be compressed with Gzip.

1. Download the latest version of BBTools from: *https://sourceforge.net/projects/bbmap/*.
2. Unpack the tar.gz file (we are using BBTools 39.43 as an example):

```
tar -xvzf BBMap_39.43.tar.gz
```

This creates a subfolder named bbmap with the shell scripts and other necessary files.

```
cd bbmap
```

3. Remove adapter sequences using BBDuk for both raw shotgun and Hi-C reads.
   a. For raw shotgun (WGS) reads:

```
./bbduk.sh in1=WGS_1.fastq in2=WGS_2.fastq out1=WGS_1_AQ.fastq.gz out2=
WGS _2_AQ.fastq.gz ref=resources/adapters.fa ktrim=r k=23 mink=11
hdist=1 minlen=50 tpe tbo
```

   b. For raw Hi-C reads:

```
./bbduk.sh in1=Hi-C_1.fastq in2=Hi-C_2.fastq out1=Hi-C_1_AQ.fastq.gz
out2=Hi-C _2_AQ.fastq.gz ref=resources/adapters.fa ktrim=r k=23 mink=11
hdist=1 minlen=50 tpe tbo
```

4. Quality trim shotgun and Hi-C reads.
   a. For shotgun (WGS) reads:

```
./bbduk.sh in1=WGS_1_AQ.fastq.gz in2=WGS_2_AQ.fastq.gz
out1=WGS_1_CL.fastq.gz out2=WGS_2_CL.fastq.gz trimq=10 qtrim=r ftm=5
minlen=50
```

   b. For Hi-C reads:

```
./bbduk.sh in1=Hi-C_1_AQ.fastq.gz in2=Hi-C_2_AQ.fastq.gz out1=Hi-C_1_
CL.fastq.gz out2=Hi-C_2_CL.fastq.gz trimq=10 qtrim=r ftm=5 minlen=50
```

5. Trim the first 10 nucleotides of Hi-C reads:

```
./bbduk.sh in1=Hi-C_1_CL.fastq.gz in2=Hi-C_2_CL.fastq.gz out1=Hi-
C_1_trim.fastq.gz out2=Hi-C_2_trim.fastq.gz ftl=10
```

6. Remove identical PCR optical and tile-edge duplicates for Hi-C reads:

```
./clumpify.sh in1=Hi-C_1_trim.fastq.gz in2=Hi-C_2_trim.fastq.gz out1=Hi-
C_1_dedup.fastq.gz out2=Hi-C_2_dedup.fastq.gz dedupe
```

7. Download pre-built binaries version of MEGAHIT (we are using MEGAHIT v1.2.9 as an example):

```
wget https://github.com/voutcn/megahit/releases/download/v1.2.9/MEGAHIT-
1.2.9-Linux-x86_64-static.tar.gz
tar zvxf MEGAHIT-1.2.9-Linux-x86_64-static.tar.gz
cd MEGAHIT-1.2.9-Linux-x86_64-static/bin/
```

8. Assemble shotgun reads using MEGAHIT:

```
./megahit -1 WGS_1_CL.fastq.gz -2 WGS_2_CL.fastq.gz -o ASSEMBLY --min-
contig-len 1000 --k-min 21 --k-max 141 --k-step 12 --merge-level 20,0.95
```

   a. Inputs: WGS_1_CL.fastq.gz and WGS_2_CL.fastq.gz, i.e., cleaned WGS read pairs from BBDuk (step 4).
   b. Outputs: The assembled contigs are written to ASSEMBLY/final.contigs.fa.

9. Install BWA in the conda environment:

```
conda create -n BWA -c conda-forge -c bioconda bwa
conda activate BWA
```

10. Align Hi-C reads to assembled contigs using BWA.
   a. Index reference sequences (assembled contigs) in the FASTA format:

```
bwa index final.contigs.fa
```

   b. Align Hi-C reads to assembled contigs:

```
bwa mem -5SP final.contigs.fa Hi-C_1_dedup.fastq.gz Hi-C_2_dedup.fastq.gz
> MAP.sam
```

      i. Inputs:
         final.contigs.fa, i.e, assembled contigs from MEGAHIT (step 8)
         Hi-C_1_dedup.fastq.gz and Hi-C_2_dedup.fastq.gz, i.e., cleaned Hi-C read pairs (step 6).
      ii. Outputs:
         Alignment file MAP.sam.

   c. Use SAMtools (H. Li et al., 2009) to remove unmapped reads, supplementary alignments, and secondary alignments:

```
samtools view -F 0x904 -bS MAP.sam > MAP_UNSORTED.bam
```

   d. Sort the BAM file by name:

```
samtools sort -n MAP_UNSORTED.bam -o MAP_SORTED.bam
```

11. Identify viral contigs using geNomad.
   a. Install geNomad through conda environment:

```
conda create -n genomad -c conda-forge -c bioconda genomad
conda activate genomad
Download the database:
genomad download-database.
```

   b. Run geNomad:

```
genomad end-to-end --cleanup --splits 8 final.contigs.fa geno-
mad_output genomad_db
```

      i. Inputs:
         final.contigs.fa, i.e., assembled contigs from MEGAHIT (step 8)
         genomad_db, i.e., predownloaded geNomad database.
      ii. Outputs:
         genomad_output/final_contigs_summary/final_contigs_virus_summary.tsv.

VIRAL GENOME BINNING AND VIRUS–HOST INTERACTION INFERENCE USING ViralCC

BASIC PROTOCOL 2

After preprocessing metagenomic Hi-C data, ViralCC (Du et al., 2023) can be used to construct the Hi-C contact matrix, retrieve draft viral metagenome-assembled genomes

(vMAGs), and detect virus–host links. ViralCC not only considers the Hi-C interaction graph but also puts forward a novel host proximity graph of viral contigs as a complementary source of information to the remarkably sparse Hi-C interaction map. The two graphs are then integrated together, followed by the Leiden graph clustering (Traag et al., 2019) using the integrative graph to generate draft viral genomes. This protocol describes how to install ViralCC and execute the full workflow for viral genome reconstruction and host association analysis.

Necessary Resources

Hardware

A 64-bit Linux system with sufficient RAM to support the in-memory operations

The memory and disk consumption heavily depend on the input data, and in many cases cannot be estimated beforehand.

Software

Python
Conda
ViralCC package

Files

Metagenomics assembled contigs (final.contigs.fa)
Hi-C alignment BAM file (MAP_SORTED.bam)
Viral contig txt file

1. Clone the repository with git:

   ```
   git clone https://github.com/dyxstat/ViralCC.git
   ```

   This creates a folder named ViralCC/ containing the source code and example test data.

2. Enter the ViralCC folder:

   ```
   cd ViralCC
   ```

3. Create the conda environment.
   a. On Linux:

      ```
      conda env create -f viralcc_linux_env.yaml
      ```

   b. On macOS:

      ```
      conda env create -f viralcc_osx_env.yaml
      ```

4. Activate the ViralCC environment:

   ```
   conda activate ViralCC_env
   ```

5. Verify the installation using the included test dataset:

   ```
   python ./viralcc.py pipeline -v Test/final.contigs.fa Test/MAP_SORTED.bam
   Test/viral_contigs.txt Test/out_test
   ```

6. Run ViralCC “raw” module to generate Hi-C contact matrix. The specific inputs and outputs are detailed in Table 1.

   ```
   python ./viralcc.py raw final.contigs.fa MAP_SORTED.bam out_directory -
   e MluCI -e Sau3AI
   ```

7. Run ViralCC “bin” module to reconstruct viral bins. The specific inputs and outputs are detailed in Table 2.

Table 1 Inputs and Outputs of the ViralCC "raw" Module

Name	Description
Inputs	
final.contigs.fa	Assembled contigs from MEGAHIT
MAP_SORTED.bam	Hi-C read alignment BAM file
–enzyme (-e)	Case-sensitive enzyme name; use multiple times for multiple enzymes
out_directory	Output directory of your choice
Outputs	
contig_info.csv	Tab-separated table including information of assembled contigs with three columns (contig name, the number of restriction sites on contigs, and contig length)
Raw_contact_matrix.npz	A sparse matrix of raw Hi-C contact maps in csr format and can be reloaded using Python command: `scipy.sparse.load_npz('Raw_contact_matrix.npz')`
ViralCC.log	log file of ViralCC

Table 2 Inputs and Outputs of the ViralCC "bin" Module

Name	Description
Inputs	
final.contigs.fa	Assembled contigs from MEGAHIT
MAP_SORTED.bam	Hi-C read alignment BAM file
viral_contig.txt	A txt file containing the names of identified viral contigs in one column without header
out_directory	Output directory of your choice
Outputs	
VIRAL_BIN	Folder containing the fasta files of draft viral bins
cluster_viral_contig.txt	Clustering results with 2 columns: the first is the viral contig name and the second is the group number
viral_contig_info.csv	Information of viral contigs with three columns (contig name, contig length, and GC-content)
prokaryotic_contig_info.csv	Information of non-viral contigs with three columns (contig name, contig length, and GC-content)
ViralCC.log	log file of ViralCC

a. Generate viral contig file in txt format containing the names of identified viral contigs:

```
python ./scripts/extract_viral_contigs.py --input /path/to/geNomad_output/final_contigs_summary/final_contigs_virus_summary.tsv --output viral_contig.txt
```

b. Run ViralCC 'bin' module:

```
python ./viralcc.py bin final.contigs.fa MAP_SORTED.bam viral_contig.txt out_directory
```

8. Run ViralCC "link" module to identify virus–host linkages from Hi-C contacts. The specific inputs and outputs are detailed in Table 3.

Table 3 Inputs and Outputs of the ViralCC "link" Module

Name	Description
Inputs	
–contig-info contig_info.csv	Information of assembled contigs with three columns (contig name, the number of restriction sites on contigs, and contig length)
–contact Raw_contact_matrix.npz	A sparse matrix of raw Hi-C contact maps in csr format and can be reloaded using Python command: `scipy.sparse.load_npz`
–viral-bin VIRAL_BIN	Folder of viral bins (FASTA; *.fa/*.fasta)
–host-bin HOST_BIN	Folder of host bins (FASTA; *.fa/*.fasta)
–viral-annotation results.csv (optional)	Virgo results.csv: A CSV giving taxonomy for each viral bin (or contig), with identifiers and an assigned lineage; use it to add viral taxonomy to your linkage results
–host-annotation gtdbtk.bac120.summary.tsv (optional)	GTDB-Tk gtdbtk.bac120.summary.tsv: A tab-delimited table mapping each host bin (user_genome) to its GTDB taxonomy (classification); attach this taxonomy to host contigs via their bin
out_directory	Output directory of your choice
Outputs	
virus_host_linkages.tsv	Tab-separated table: viral contig, host contig, contact strength, viral annotation (optional), and host annotation (optional)
ViralCC.log	log file of ViralCC

```
python ./viralcc.py link --contact Raw_contact_matrix.npz --contig-
info contig_info.csv --viral-bin out_directory/VIRAL_BIN --host-
bin /path/to/ImputeCC_Output/HOST_BIN --viral-annotation /path/to/
Virgo_Output/results.csv --host-annotation /path/to/GTDBTK_Output/
gtdbtk.bac120.summary.tsv out_directory
```

SUPPORT PROTOCOL 1

HOST GENOME BINNING USING ImputeCC

ImputeCC (Du et al., 2024) is an integrative contig-binning tool designed specifically for metaHi-C datasets. It combines Hi-C interaction information with the inherent discriminative power of single-copy marker genes, first generating preliminary bins and then refining them using a constrained random walk with restart (CRWR) algorithm to enhance Hi-C connectivity among contigs. Before running ImputeCC, raw Hi-C contacts must be normalized using NormCC, the normalization module implemented in the MetaCC framework (Du et al., 2023), which corrects systematic biases in Hi-C contacts and outputs a normalized Hi-C contact matrix together with contig-level information. This protocol describes how to: (i) generate NormCC-normalized Hi-C contacts and contig information, and (ii) run the ImputeCC binning workflow. Both NormCC and ImputeCC are freely available as open-source tools on GitHub.

Necessary Resources

Hardware

A 64-bit Linux system with sufficient RAM to support the in-memory operations

Software

Python
Conda

Table 4 Inputs and Outputs of the NormCC Normalization Module

Name	Description
Inputs	
final.contigs.fa	Assembled contigs from MEGAHIT (Basic Protocol 1)
MAP_SORTED.bam	Hi-C read alignment file sorted by read name (Basic Protocol 1)
Outputs	
contig_info.csv	Contig information table with three columns: contig name, number of restriction sites on the contig, and contig length
Normalized_contact_matrix.npz	Sparse matrix of normalized Hi-C contact maps in SciPy CSR format (can be reloaded with `scipy.sparse.load_npz`)

NormCC (see installation steps below)
ImputeCC (see installation steps below)

Files

Metagenomics assembled contigs (final.contigs.fa)
Normalized Hi-C contact matrix (Normalized_contact_matrix.npz)
A contig information file (contig_info.csv)

Generate a NormCC-normalized Hi-C contact matrix (NormCC module in MetaCC)

1. Clone the MetaCC repository with git (if not already installed):

```
git clone https://github.com/dyxstat/MetaCC.git
cd MetaCC
```

2. Create the MetaCC conda environment:

```
conda env create -f MetaCC_env.yaml
conda activate MetaCC_env
```

3. Run the NormCC normalization module to normalize raw Hi-C contacts (see Table 4 for input and output details):

```
python ./MetaCC.py norm -e MluCI -e Sau3AI -v final.contigs.fa MAP_SORTED.bam
out_NormCC
```

Host genome binning using ImputeCC

4. Clone the repository with git:

```
git clone https://github.com/dyxstat/ImputeCC.git
```

5. Enter the ImputeCC folder:

```
cd ImputeCC
```

6. Activate the conda environment:

```
conda activate ImputeCC_env
```

7. Run the ImputeCC binning pipeline (see Table 5 for input and output details):

```
python ./ImputeCC.py pipeline -v final.contigs.fa contig_info.csv Normal-
ized_contact_matrix.npz out_directory
```

Table 5 Inputs and Outputs of the ImputeCC Binning Pipeline

Name	Description
Inputs	
final.contigs.fa	Assembled contigs from MEGAHIT (Basic Protocol 1)
contig_info.csv	Contig information file generated by NormCC
Normalized_contact_matrix.npz	Normalized Hi-C contact matrix generated by NormCC
out_directory	Output directory of your choice
Outputs	
FINAL_BIN/	Folder containing the fasta files of draft genomic bins
ImputeCC.log	Specific implementation information of ImputeCC

SUPPORT PROTOCOL 2

HOST MAG TAXONOMIC ANNOTATION WITH GTDB-Tk

Taxonomic profiling is a critical downstream analysis in metagenomics, enabling the identification of species and the exploration of microbial community composition. This protocol describes how to annotate host metagenome-assembled genomes (MAGs) using GTDB-Tk (Chaumeil et al., 2019), a reference-based taxonomy classification tool that assigns standardized GTDB lineages to genome bins. The resulting annotations assign standardized taxonomy to host MAGs, facilitating interpretation of microbial composition and linking potential virus–host pairs.

Necessary Resources

Hardware

A 64-bit Linux system with sufficient RAM to support the in-memory operations

For archaea, ∼60 GB memory and ∼140 GB storage; for bacteria, ∼110 GB memory and ∼140 GB storage.

Software

Python
Conda

Files

HOST_BIN/ [directory containing the host MAGs (FASTA files)]

1. Create the GTDB-tk conda environment:

```
conda create -n gtdbtk-2.5.2 -c conda-forge -c bioconda gtdbtk=2.5.2
```

Replace 2.5.2 with the version you want to install.

2. Download the reference data. The conda package is bundled with a script download-db.sh that will automatically download and extract the GTDB-Tk reference data.

```
download-db.sh
```

3. Activate the conda environment:

```
conda activate gtdbtk-2.5.2
```

4. Run GTDB-tk with the following command (see Table 6 for input and output details):

```
gtdbtk classify_wf --genome_dir VIRAL_BIN --out_dir gtdbtk_output --
extension fa
```

Table 6 Inputs and Outputs of the GTDB-tk[a]

Name	Description
Inputs	
HOST_BIN/	Directory containing host MAG FASTA files (Support Protocol 1)
Outputs	
gtdbtk.[domain].summary.tsv	Summary of bacterial or archaeal genome classifications
gtdbtk.json	The console output of GTDB-Tk saved to disk in a JSON format
gtdbtk.log	The console output of GTDB-Tk saved to disk
gtdbtk.warnings.log	The verbose output of any GTDB-Tk warnings which were encountered; if this file is written to, then the main GTDB-Tk console will highlight this as a warning message
align/	Contains MSA of the submitted genomes
classify/gtdbtk.backbone.[domain].classify.tree	Reference tree in Newick format containing query genomes placed with pplacer
classify/gtdbtk.[domain].tree.mapping.tsv	Map between genomes and the class_level taxonomy tree used to classify them
classify/gtdbtk.[domain].classify.tree.[index].tree	Reference tree in Newick format containing query genomes placed with pplacer
identify/gtdbtk.[domain].markers_summary.tsv	A summary of unique, duplicated, and missing markers within the bac120 marker set, or the ar53 marker set for each submitted genome
identify/gtdbtk.translation_table_summary.tsv	A summary of translation tables determined for each genome
identify/gtdbtk.failed_genomes.tsv	File reporting failed genomes which have been excluded from analysis due to Prodigal failing to call any genes

[a] The [domain] placeholder in the output file names is replaced by bac120 (for bacterial marker genes) or ar53 (for archaeal marker genes).

VIRAL BIN TAXONOMIC ANNOTATION WITH Virgo

SUPPORT PROTOCOL 3

After reconstructing viral bins from metagenomic Hi-C data, a key downstream step is to assign taxonomic labels to each viral genome. This protocol describes how to annotate viral bins using Virgo (Riccardi et al., 2025), a taxonomy classification framework that integrates genomic marker profiles with Jaccard similarity to assign standardized ICTV-based taxonomic lineages. The resulting taxonomy provides biological context for the reconstructed viral genomes and enables comparative analyses across viral communities.

Necessary Resources

Hardware

A 64-bit Linux system with sufficient RAM to support the in-memory operations

The Virgo reference database requires ∼1.3 GB of disk space.

Software

Python
Conda

Files

VIRAL_BIN/ [directory containing viral MAGs (FASTA files) generated from ViralCC]

Table 7 Inputs and Outputs of the Virgo

Name	Description
Inputs	
VIRAL_BIN/	Folder containing the FASTA files of draft viral bins generated by ViralCC
VMR_MSL40v1_VirgoDB/	Downloaded Virgo database (the version may vary)
Outputs	
results.csv	Taxonomy assignments for all viral bins

1. Clone the repository and navigate to the project directory:
   ```
   git clone https://github.com/christopher-riccardi/Virgo.git
   cd Virgo
   ```
2. Create and activate the environment using the environment.yaml file:
   ```
   conda env create -f environment/environment.yaml
   conda activate virgo
   ```
3. Download and unpack the Virgo database.
   a. Download the appropriate Virgo reference database (e.g., ICTV MSL40 v1) from: *https://drive.google.com/drive/u/0/folders/1fUnYjDS032cuenEzN3K7RArV55KCo8eD*
   b. Unzip the downloaded file:
      ```
      unzip VMR_MSL40v1_VirgoDB.zip
      ```
4. Run Virgo to annotate viral bins (see Table 7 for input and output details):
   ```
   python src/virgo.py -i VIRAL_BIN -o Virgo_output -d VMR_MSL40v1_VirgoDB/ -t 16 --with-replacement
   ```

SUPPORT PROTOCOL 4

VISUALIZATION OF VIRUS–HOST INTERACTION NETWORKS WITH MetaHiCNet

After reconstructing viral and host metagenome-assembled genomes (MAGs) and annotating their taxonomy, an important downstream step is to explore the interaction structure of the community. MetaHiCNet (Sun et al., 2025) is an open-access web server that provides a stepwise workflow for the interactive visualization of microbial interaction networks, including taxonomic treemaps, cross–taxa networks, and cross–bin (MAG-level) networks. In this support protocol, we describe how to: (i) prepare input files for MetaHiCNet from the outputs of ViralCC, ImputeCC, GTDB–Tk, and Virgo; and (ii) use MetaHiCNet to visualize normalized Hi–C interaction networks at both taxonomic and bin levels. MetaHiCNet is freely accessible at: *https://metahicnet.com* without login.

Necessary Resources

Hardware

A computer with a modern web browser and a stable internet connection

Software

A modern web browser (Chrome, Firefox, Safari, or Edge)

Files

Contig Information File (tab–delimited with columns: contig ID, number of restriction sites, contig length), derived from ViralCC's raw module

Raw Hi–C Contact File listing pairwise Hi–C contact counts between contigs (e.g., Raw_contact_matrix.npz produced by ViralCC's raw module)

Binning Information File mapping each contig to a bin ID (e.g., host bins from ImputeCC FINAL_BIN/ and viral bins from ViralCC VIRAL_BIN/)

Taxonomy Information File assigning taxonomic lineages to bins or contigs, constructed from GTDB–Tk outputs for host MAGs and Virgo results for viral bins

Prepare input files for MetaHiCNet

1. Construct the Contig Information File.
   a. Start from the contig_info.csv file generated by NormCC (Support Protocol 1) or from an equivalent table produced by ViralCC's raw module.
   b. Ensure that the file includes at least the following columns for each contig:
      contig ID (matching the FASTA headers in final.contigs.fa)
      number of restriction sites on the contig
      contig length (bp)
      contig coverage (if available).
   c. Save the file in a tab–delimited text format recognized by MetaHiCNet (e.g., contig_info_for_MetaHiCNet.tsv).

2. Construct the Binning Information File (optional but required for cross–bin visualization).
   a. For each host bin FASTA file in FINAL_BIN/ (ImputeCC; Support Protocol 1), parse the contig headers and assign a unique bin ID (e.g., HostBin_001, HostBin_002, …).
   b. For each viral bin FASTA file in VIRAL_BIN/ (ViralCC; Basic Protocol 2), parse the contig headers and assign bin IDs (e.g., ViralBin_001, ViralBin_002, …).
   c. Combine these into a single tab–delimited table saved as binning_info.tsv with at least two columns:
      contig_id, i.e., contig name
      bin_id, i.e., bin name (host or viral).

3. Construct the Taxonomy Information File (optional but required for taxonomic treemaps and cross–taxa networks).
   a. From GTDB–Tk outputs (Support Protocol 2), extract taxonomy for each host bin from gtdbtk.bac120.summary.tsv and gtdbtk.ar53.summary.tsv, using the user_genome column as the bin ID.
   b. From Virgo's results.csv (Support Protocol 3), extract taxonomy for each viral bin (e.g., ICTV–based lineage).
   c. Merge host and viral annotations into a single tab–delimited file saved as taxonomy_info.tsv. with columns such as:
      Bin_id
      domain, phylum, class, order, family, genus, species (GTDB or ICTV ranks)
      optional fields (e.g., genome completeness, quality flags).

4. Construct the Raw Hi–C Contact File.
   a. Start from Raw_contact_matrix.npz generated by ViralCC's raw module (Basic Protocol 2, Table 1), which is a sparse matrix of raw Hi–C contact maps.
   b. Use a short Python/SciPy script to convert this matrix to a long–format table with 3 columns:
      contig_i, i.e., row contig ID
      contig_j, i.e., column contig ID
      raw_contact, i.e., integer Hi–C contact count between contig_i and contig_j (exclude entries with zero contacts).

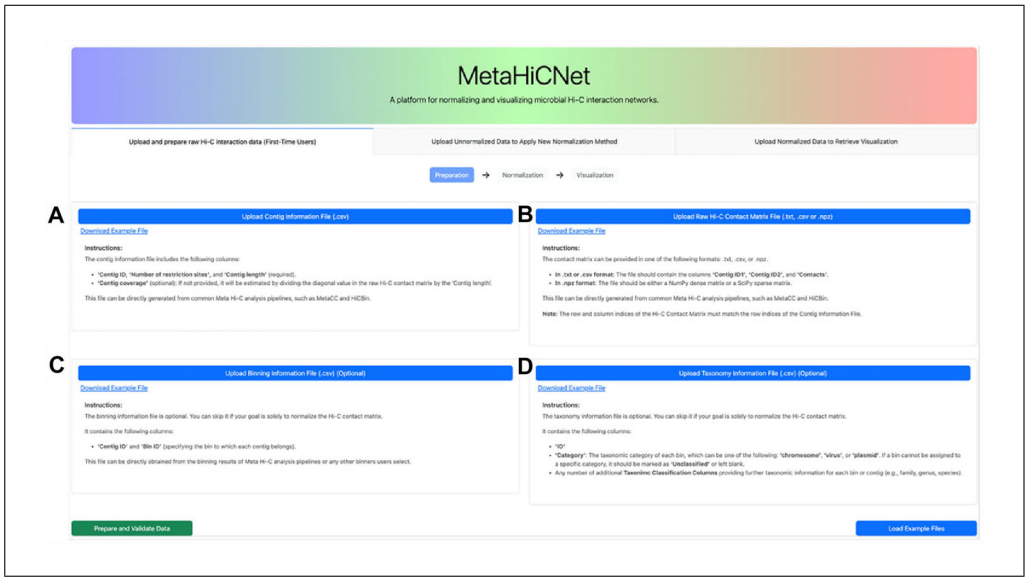

Figure 1 MetaHiCNet data preparation interface. Users upload input files prior to visualization. The four panels accept: (**A**) a required contig information file containing contig ID, number of restriction sites, and contig length; (**B**) a required raw Hi-C contact matrix in NPZ, csv, or txt format; and optional (**C**) binning and (**D**) taxonomy files for MAG- and taxon-level visualizations. Example templates can be downloaded from each panel, and the "Prepare and Validate Data" button initiates input checking before downstream MetaHiCNet analysis.

c. Ensure that contig IDs exactly match those used in the Contig Information and Binning Information files.
d. Save the table as raw_hic_contacts.tsv.

Visualize interaction networks with MetaHiCNet

5. Once all input files are prepared, open the MetaHiCNet web server in a modern browser, upload the contig, binning, taxonomy, and raw Hi–C contact files (Fig. 1), and run the visualization modules. MetaHiCNet will generate interactive cross–bin or cross–taxa Hi–C networks, allowing users to inspect virus–host connectivity and export publication–ready figures based on normalized interaction strengths.

COMMENTARY

Background Information

Metagenomic Hi-C extends chromosome conformation capture to mixed microbial communities, preserving the physical proximity of DNA fragments through cross-linking and ligation before sequencing. When applied to environmental or host-associated microbiomes, metaHi-C yields contact maps in which contigs from the same cell show elevated Hi-C linkage, enabling improved metagenomic binning and resolving plasmid and phage associations that are difficult to detect from shotgun data alone. Recent computational tools now exploit Hi-C for virus–host linkage inference. ViralCC constructs an interaction graph between viral and host contigs, incorporates a host-proximity graph to mitigate Hi-C sparsity, and applies graph clustering to recover viral bins. ImputeCC is tailored for Hi-C–informed host genome binning by combining normalized Hi-C contacts with single-copy marker genes and using a constrained random walk with restart to improve host MAG completeness and purity. NormCC addresses a central challenge in metaHi-C, strong biases from restriction-site density, contig length, and coverage, by modeling and removing these effects to produce normalized contact matrices suitable for downstream inference. Taxonomic annotation and visualization are essential for biological interpretation of these outputs. GTDB-Tk provides standardized, genome-based taxonomy for bacterial and archaeal MAGs, while Virgo assigns ICTV-based taxonomy to viral bins using genomic marker profiles and similarity-based approaches, converting draft viral bins into taxa that can be placed in ecological and evolutionary context. MetaHiCNet then integrates outputs from ViralCC, ImputeCC,

GTDB-Tk, and Virgo in a web-based environment for generating cross-taxa or cross-bin Hi-C networks that highlight prominent virus–host interactions. Compared with strategies relying solely on sequence similarity, CRISPR spacer matches, or co-abundance, this integrated workflow uses direct physical proximity signals, improves viral and host genome recovery, provides harmonized taxonomy for both domains, and yields interpretable network visualizations.

Critical Parameters

Several protocol choices warrant special attention. Adequate WGS and Hi-C depth is fundamental: poor WGS coverage yields fragmented assemblies that lower the resolution of Hi-C-based binning and virus–host inference, while inadequate Hi-C depth produces sparse contact matrices that obscure true interactions. Library preparation and restriction-enzyme choice (e.g., MluCI, Sau3AI) determine restriction-site density and distribution; keeping the preparation consistent across samples is important for comparability. Assembly quality also matters: very short contigs usually contain few restriction sites and generate noisy Hi-C signal, so ViralCC, NormCC, and ImputeCC default to a minimum contig length of 1000 bp. In ViralCC, the thresholds for constructing the host-proximity graph, the clustering resolution, and the filtering of low-quality interactions strongly influence the number and quality of inferred vMAGs and virus–host links. Normalization (NormCC) and downstream binning/annotation also require careful tuning. Effective bias correction depends on accurate contig metadata (restriction-site counts, length, coverage); errors in contig_info.csv can cause incorrect correction, and noticeable residual correlations between normalized contacts and bias factors indicate parameters should be revisited. The constrained random walk in ImputeCC depends on parameters, including restart probabilities, maximum iterations, convergence criteria, and thresholds to merge contigs with single-copy marker genes: overly lenient thresholds can group unrelated contigs, whereas overly strict thresholds may leave many contigs unbinned. GTDB-Tk and VIRGO rely on versioned reference databases; mixing releases across samples complicates comparisons, so always record the database and release used. Finally, MetaHiCNet visualizations are highly sensitive to thresholds for normalized contact strength and node/edge filtering. Therefore, users should treat them as exploratory and report the exact filters used in figures.

Troubleshooting

The pipeline described above has several intermediate steps that influence the final outputs. When issues arise, we recommend first carefully reviewing the log files generated by each tool, as they often contain informative error messages or warnings. In addition, we summarize common problems, their likely causes, and recommended solutions in Table 8.

Understanding Results

In a successful run of these protocols, users should obtain: (i) well-processed shotgun and Hi-C reads ready for assembly and alignment, a high-quality shotgun assembly, and a BAM file of Hi-C reads mapped to the assembly; (ii) Hi-C contact matrices in which raw contact counts are initially correlated with contig length, restriction-site count, and coverage, but these correlations are substantially reduced after NormCC normalization; (iii) coherent sets of viral bins with high completeness evaluated by CheckV (Nayfach et al., 2021a) and host bins with reasonable completeness and low contamination evaluated by CheckM2 (Chklovski et al., 2023); and (iv) GTDB-Tk and Virgo taxonomies that place most MAGs and vMAGs into plausible clades.

In MetaHiCNet, well-normalized data should yield interaction networks in which strong edges concentrate within host bins and between virus–host pairs. Taken together, these results allow users to determine which viral and host genomes are present, how they are taxonomically distributed, and which virus–host pairs are most strongly supported by Hi-C contacts. By contrast, several patterns indicate unreliable results and should prompt re-examination of earlier steps: very low mapping rates or extremely fragmented assemblies; NormCC outputs in which normalized contact strengths remain strongly correlated with contig length or restriction-site count; bin sets dominated by highly incomplete or heavily contaminated MAGs; GTDB-Tk and Virgo returning mostly "unclassified" assignments or taxonomies that contradict the sample context; or MetaHiCNet networks that show no edges at reasonable thresholds or are dominated by a few unrealistic "hub" nodes connected to nearly all contigs. In such cases, apparent virus–host links and network structures should not be over-interpreted until library quality,

Table 8 Troubleshooting Guide to Potential Errors When Executing the Protocols

Problem	Possible cause	Solution
Assembled contigs are highly fragmented and short	Insufficient WGS sequencing depth; overly stringent assembly parameters; presence of many low-abundance taxa	Increase WGS depth where possible; relax MEGAHIT parameters; optionally perform co-assembly across related samples; remove very low-coverage contigs before downstream analyses
Very few Hi-C contacts are observed between contigs	Low Hi-C library complexity; over-stringent read mapping or filtering; high proportion of unmapped reads	Review Hi-C library preparation; relax mapping parameters slightly while maintaining quality; ensure the Hi-C library and the shotgun library come from the same sample
ViralCC recovers very few vMAGs or virus–host links	Viral contigs are under-represented or too short; contact matrix is too sparse; thresholds for clustering or link calling are too strict	Check geNomad output for number and length distribution of viral contigs; lower minimal thresholds in ViralCC; verify that Raw_contact_matrix.npz and contig_info.csv match the same assembly
ImputeCC produces many small or low-quality host bins	Weak Hi-C signal between contigs; inaccurate or incomplete contig_info.csv; insufficient single-copy marker genes per bin	Confirm that NormCC normalization completed successfully; check contig_info.csv for missing or inconsistent entries; adjust ImputeCC parameters; consider filtering very short or low-coverage contigs before binning
GTDB-Tk fails or returns 'unclassified' for many host MAGs	Bins are incomplete or contaminated; genome quality thresholds not met	Evaluate MAG completeness and contamination with CheckM2 or similar tools; merge or split bins as needed; confirm that the correct GTDB reference data are downloaded and that GTDB-Tk is up to date
Virgo results.csv contains few or no taxonomic assignments	Viral bins are too fragmented or divergent from the reference database; incorrect database version or path; insufficient marker detection	Verify that VIRAL_BIN/ contains non-empty FASTA files; confirm the database path in the Virgo command; consider using the latest ICTV database; relax confidence thresholds if available
MetaHiCNet visualizations are empty or show no edges	Mismatched contig or bin IDs across input files; overly strict thresholds for normalized contacts; incorrect file formats	Confirm that contig IDs are consistent across Contig, Binning, Taxonomy, and Contact files; inspect a few lines manually; reduce contact strength thresholds in MetaHiCNet; ensure files are tab-delimited and follow the expected column names

normalization settings, and binning parameters have been reviewed and, if necessary, corrected.

Acknowledgements

This project is supported by NSF grant EF-2125142 to F.S. Y.D. is partially supported by the University of Texas Systems STARs Program.

Author Contributions

Yuxuan Du: Conceptualization; methodology; supervision; visualization; writing—original draft; writing—review and editing. **Yuqiu Wang**: Data curation; methodology; software; validation; writing—original draft; writing—review and editing. **Fengzhu Sun**: Conceptualization; methodology; project administration; supervision; writing—review and editing.

Conflict of Interest

The authors declare that they have no competing interests.

Data Availability Statement

Tutorial-scale example files supporting this protocol are available at: *https://doi.org/10.5281/zenodo.18461835*.

Literature Cited

Bushnell, B. (2014). BBMap: A Fast, Accurate, Splice-Aware Aligner. Osti.gov. https://www.osti.gov/biblio/1241166

Camargo, A. P., Roux, S., Schulz, F., Babinski, M., Xu, Y., Hu, B., Chain, P. S. G., Nayfach, S., & Kyrpides, N. C. (2023). Identification of mobile genetic elements with geNomad. *Nature Biotechnology*, 1–10. https://doi.org/10.1038/s41587-023-01953-y

Chaumeil, P.-A., Mussig, A. J., Hugenholtz, P., & Parks, D. H. (2019). GTDB-Tk: A toolkit to classify genomes with the Genome Taxonomy Database. *Bioinformatics*, *36*(6). https://doi.org/10.1093/bioinformatics/btz848

Chklovski, A., Parks, D. H., Woodcroft, B. J., & Tyson, G. W. (2023). CheckM2: A rapid, scalable and accurate tool for assessing microbial genome quality using machine learning. *Nature Methods*, *20*(8), 1203–1212. https://doi.org/10.1038/s41592-02301940-w

Du, Y., Fuhrman, J. A., & Sun, F. (2023). ViralCC retrieves complete viral genomes and virus-host pairs from metagenomic Hi-C data. *Nature Communications*, *14*(1), 502. https://doi.org/10.1038/s41467-023-35945-y

Du, Y., & Sun, F. (2023). MetaCC allows scalable and integrative analyses of both long-read and short-read metagenomic Hi-C data. *Nature Communications*, *14*(1), 6231. https://doi.org/10.1038/s41467-023-41209-6

Du, Y., Zuo, W., & Sun, F. (2024). Imputing metagenomic Hi-C contacts facilitates the integrative contig binning through constrained random walk with restart. *Journal of Computational Biology*, *31*(10), 1008–1021. https://doi.org/10.1089/cmb.2024.0663

Edwards, R. A., McNair, K., Faust, K., Raes, J., & Dutilh, B. E. (2016). Computational approaches to predict bacteriophage–host relationships. *FEMS Microbiology Reviews*, *40*(2), 258–272. https://doi.org/10.1093/femsre/fuv048

Gilbert, J. A., Jansson, J. K., & Knight, R. (2014). The Earth Microbiome project: Successes and aspirations. *BMC Biology*, *12*, 69. https://doi.org/10.1186/s12915-014-0069-1

Li, D., Liu, C.-M., Luo, R., Sadakane, K., & Lam, T.-W. (2015). MEGAHIT: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, *31*(10), 1674–1676. https://doi.org/10.1093/bioinformatics/btv033

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. ArXiv.org. https://arxiv.org/abs/1303.3997

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., & Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079.

Nayfach, S., Camargo, A. P., Schulz, F., Eloe-Fadrosh, E., Roux, S., & Kyrpides, N. C. (2021a). CheckV assesses the quality and completeness of metagenome-assembled viral genomes. *Nature Biotechnology*, *39*(5), 578–585. https://doi.org/10.1038/s41587-020-00774-7

Nayfach, S., Roux, S., Seshadri, R., Udwary, D., Varghese, N., Schulz, F., Wu, D., Paez-Espino, D., Chen, I.-M., Huntemann, M., Palaniappan, K., Ladau, J., Mukherjee, S., Reddy, T. B. K., Nielsen, T., Kirton, E., Faria, J. P., Edirisinghe, J. N., Henry, C. S., … Eloe-Fadrosh, E. A. (2021b). A genomic catalog of Earth's microbiomes. *Nature Biotechnology*, *39*(4), 499–509. https://doi.org/10.1038/s41587-020-0718-6

Nurk, S., Meleshko, D., Korobeynikov, A., & Pevzner, P. A. (2017). metaSPAdes: A new versatile metagenomic assembler. *Genome Research*, *27*(5), 824–834. https://doi.org/10.1101/gr.213959.116

Parks, D. H., Rinke, C., Chuvochina, M., Chaumeil, P.-A., Woodcroft, B. J., Evans, P. N., Hugenholtz, P., & Tyson, G. W. (2017). Recovery of nearly 8,000 metagenome-assembled genomes substantially expands the tree of life. *Nature Microbiology*, *2*(11), 1533–1542. https://doi.org/10.1038/s41564-017-0012-7

Riccardi, C., Wang, Y., Yooseph, S., & Sun, F. (2025). Bidirectional subsethood of shared marker profiles enables accurate virus classification. *Microbiome*, *13*(1), 170. https://doi.org/10.1186/s40168-025-02159-x

Roux, S., Brum, J. R., Dutilh, B. E., Sunagawa, S., Duhaime, M. B., Loy, A., Poulos, B. T., Solonenko, N., Lara, E., Poulain, J., Pesant, S., Kandels-Lewis, S., Dimier, C., Picheral, M., Searson, S., Cruaud, C., Alberti, A., Duarte, C. M., Gasol, J. M., & Sullivan, M. B. (2016). Ecogenomics and potential biogeochemical impacts of globally abundant ocean viruses. *Nature*, *537*(7622), 689–693. https://doi.org/10.1038/nature19366

Sun, C., Qin, Z., Liu, R., Guo, Y., Ge, Y., & Du, Y. (2025). MetaHiCNet: A web server for normalizing and visualizing microbial Hi-C interaction networks. *Nucleic Acids Research*, *53*(W1), W383–W389. https://doi.org/10.1093/nar/gkaf340

Thompson, L. R., Sanders, J. G., McDonald, D., Amir, A., Ladau, J., Locey, K. J., Prill, R. J., Tripathi, A., Gibbons, S. M., Ackermann, G., Navas-Molina, J. A., Janssen, S., Kopylova, E., Vázquez-Baeza, Y., González, A., Morton, J. T., Mirarab, S., Xu, Z. Z., Jiang, L., & Knight, R. (2017). A communal catalogue reveals Earth's multiscale microbial diversity. *Nature*, *551*(7681), 457–463. https://doi.org/10.1038/nature24621

Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: Guaranteeing well-connected communities. *Scientific Reports*, *9*(1), 5233. https://doi.org/10.1038/s41598-019-41695-z