# Recovering Eukaryotic Genomes and Transcriptomes using Eukfinder and Anvi'o

Dandan Zhao

Apr 24, 2025

d.zhao@dal.ca

AMERICAN SOCIETY FOR MICROBIOLOGY | mBio

Check for updates

| Environmental Microbiology | Research Article

# Eukfinder: a pipeline to retrieve microbial eukaryote genome sequences from metagenomic data

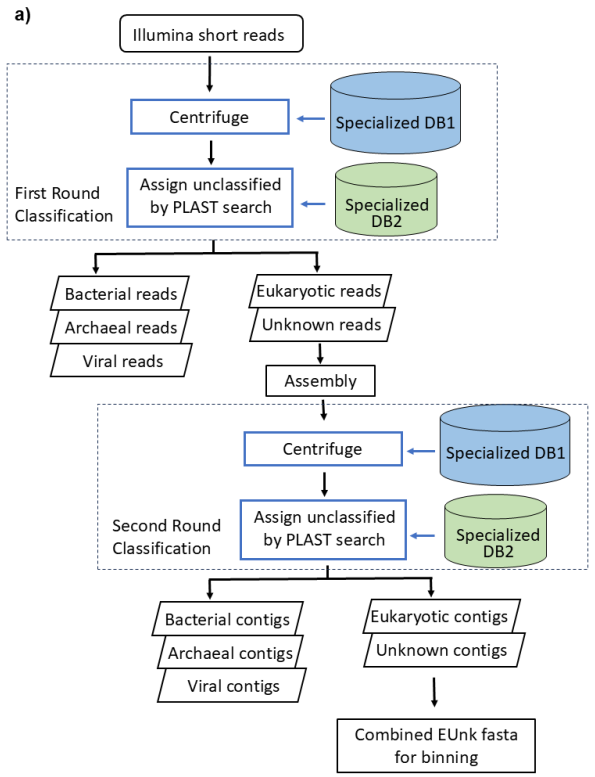
**Dandan Zhao,[1,2] Dayana E. Salas-Leiva,[1,3] Shelby K. Williams,[1,2] Katherine A. Dunn,[1,2] Jason D. Shao,[1,2] Andrew J. Roger[1,2]**

**AUTHOR AFFILIATIONS** See affiliation list on p. 17.

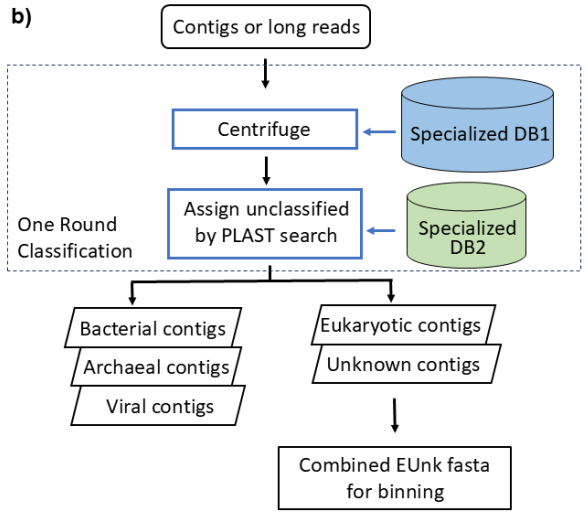
**ABSTRACT** Whole-genome shotgun (WGS) metagenomic sequencing of microbial communities enables the discovery of the functions, physiologies, and evolutionary histories of prokaryotic and eukaryotic microbes. However, metagenomic studies of microbial eukaryotes lag due to challenges in identifying and assembling high-quality genomes from WGS data. To address this problem, we developed Eukfinder, a bioinformatics pipeline that identifies potential eukaryotic sequences from WGS metagenomic data, with a complementary binning workflow for recovering nuclear and mitochon-

# Eukfinder Classification workflow

## Short Reads (Eukfinder_short)




## Long Reads or assembled contigs (Eukfinder_long)



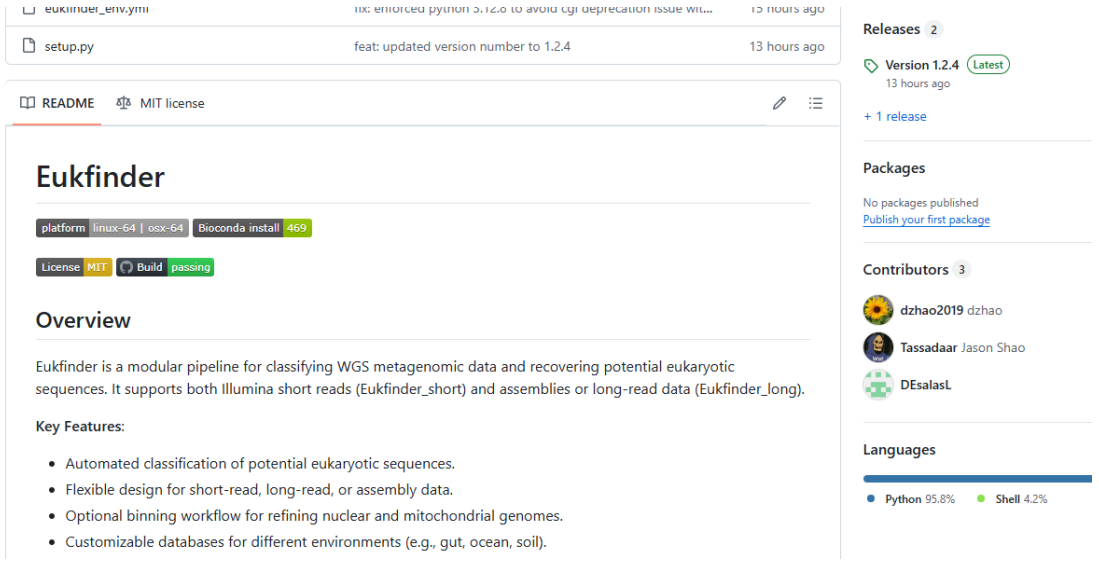

| Workflow | Input | Step Summary |
|---|---|---|
| Short | Short Reads (fastq files) | Trim* → Dehost* → Centrifuge* → PLAST → Assemble → Reclassify → EUnk.fasta for Bin |
| Long | Long Reads/Contigs | Direct classification (Centrifuge + PLAST) → EUnk.fasta for Bin |

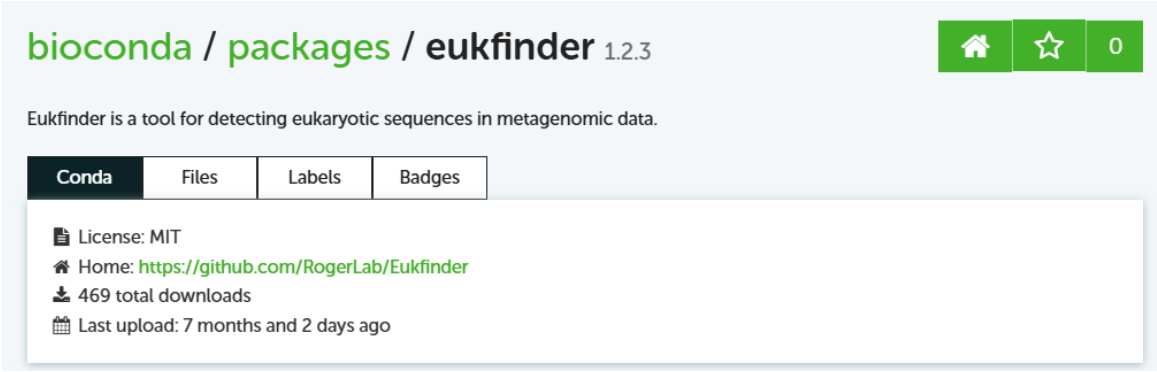
* Running in read_prep step

# Where to find Eukfinder

- Github:
  https://github.com/RogerLab/Eukfinder
- Use Guide (work in process):
  https://github.com/RogerLab/Eukfinder/wiki

- Conda installation package:
  https://anaconda.org/bioconda/eukfinder



# Where to find Eukfinder

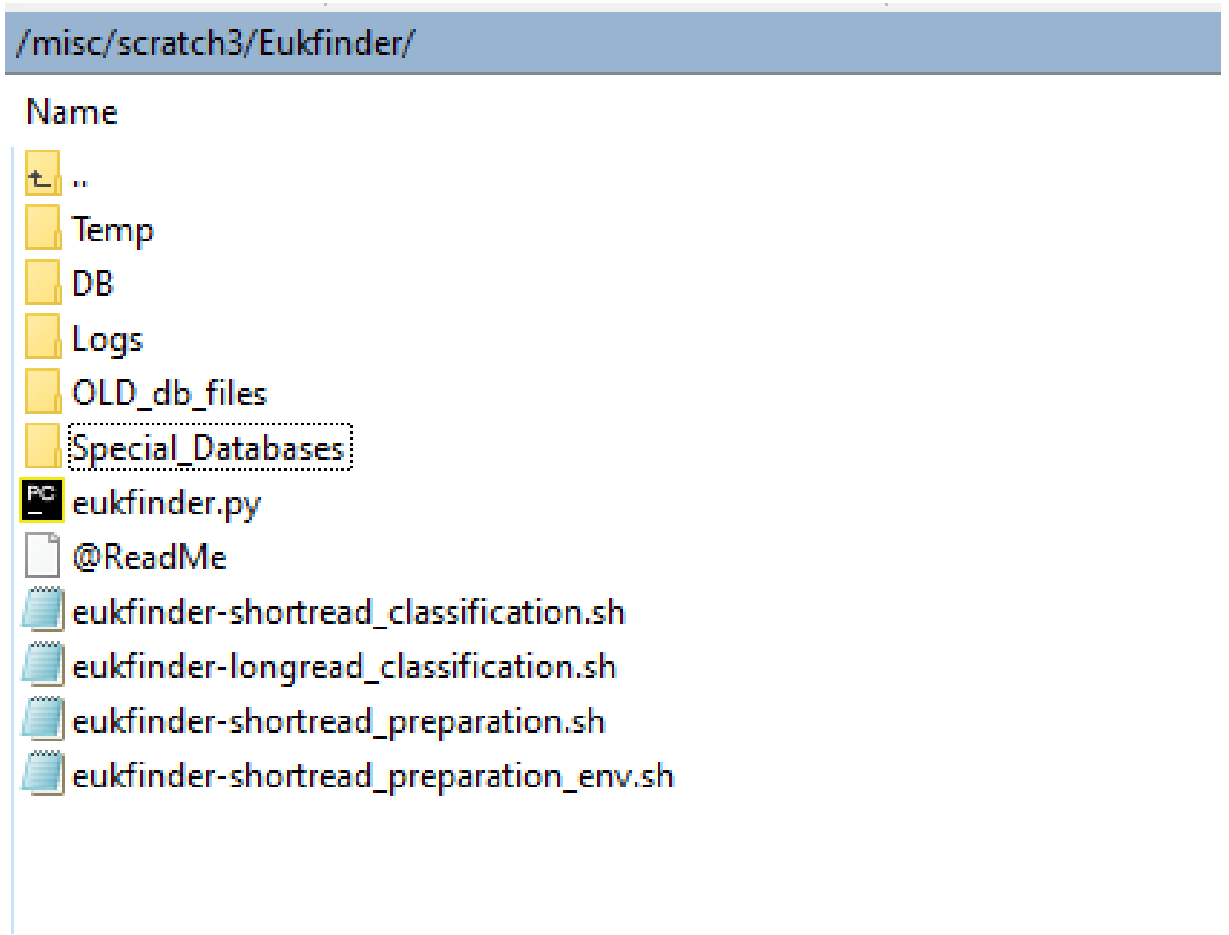
- Perun: /scratch3/Eukfinder/

  Main script: eukfinder.py

  Databases:

    DB/

    Special_Databases/



# How to run Eukfinder

- **1. Created environment and install eukfinder**

  Anaconda or miniconda required

  `conda create -n eukfinder -c bioconda eukfinder`

- **2. Download or build databases**
  - Plast Database
  - Centrifuge Database
  - Human Genome for read decontamination
  - Read Adapters for Illumina sequencing

**System Requirements**

| System | Minimum required | Recommended |
|---|---|---|
| **Disk Space** | ~200 GB | ≥ 300 GB |
| **Memory** | 32 GB | 64–128 GB |
| **Network connectivity** | Connection to Internet | High-speed connection to download the databases efficiently |

# New version is coming:

OLD version: conda package: eukfinder=1.2.3

inside Eukfinder.py

__version__ = '1.0.0'

-> https://anaconda.org/bioconda/eukfinder

-> currently installed on Perun

NEW version: 1.2.4

inside Eukfinder.py

__version__ = '1.2.4'

➔Github

https://github.com/RogerLab/Eukfinder/blob/main/bin/Eukfinder.py

➔Location on Perun

/scratch3/Eukfinder/eukfinder.py

> python3 /scratch3/Eukfinder/eukfinder.py -h

NOTE:

1. After installed via conda, download eukfinder.py from github

2. All the settings/parameters in this talk are for new version v1.2.4

# *Important Notes for v1.2.4*

1. The newest version removes the need for acc2tax.

2. Only Centrifuge_DB and PLAST_DB are required.

3. Eukfinder v1.2.4 is not yet installed globally on Perun.

*How to Run Eukfinder v1.2.4 on Perun before the installation*

```
source activate eukfinder
Eukfinder=/misc/scratch3/Eukfinder/eukfinder.py
python $Eukfinder [arguments]
```

# How to download Eukfinder databases

**source activate eukfinder**

**eukfinder.py -h**

```
usage: eukfinder [-h]
                 {short_seqs,read_prep,long_seqs,read_prep_env,download_db}
                 ...

positional arguments:
  {short_seqs,read_prep,long_seqs,read_prep_env,download_db}

optional arguments:
  -h, --help            show this help message and exit
```

**eukfinder.py download_db -h**

```
usage: eukfinder download_db [-h] [-n NAME] [-p PATH]

optional arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  directory name for storing the databases
  -p PATH, --path PATH  filesystem path for storing the databases
```

# How to download Eukfinder databases

**eukfinder.py download_db -n Eukfinder_DB**

```
Created /home/dzhao/.eukfinder/Eukfinder_DB

Would you like to download all databases (72 GB)? (yes/no)
>no

Please select database(s) which you would like to install, separated by spaces (e.g., 1 2).

1. centrifuge database - 70 GB
2. PLAST database - 2.1 GB
3. Human Genome for read decontamination - 0.92 GB
4. Read Adapters for Illumina sequencing - 2.4 KB
5. test set - 51 MB
6. test set with tiny dbs - 51 MB

Or type exit, if you would like to skip for now:
>exit

Deleted /home/dzhao/.eukfinder/Eukfinder_DB

No downloads, exiting...
```

**If choose yes,**
**it will download and uncompress all databases automatically, and you don't need to specify database locations when you run Eukfinder**

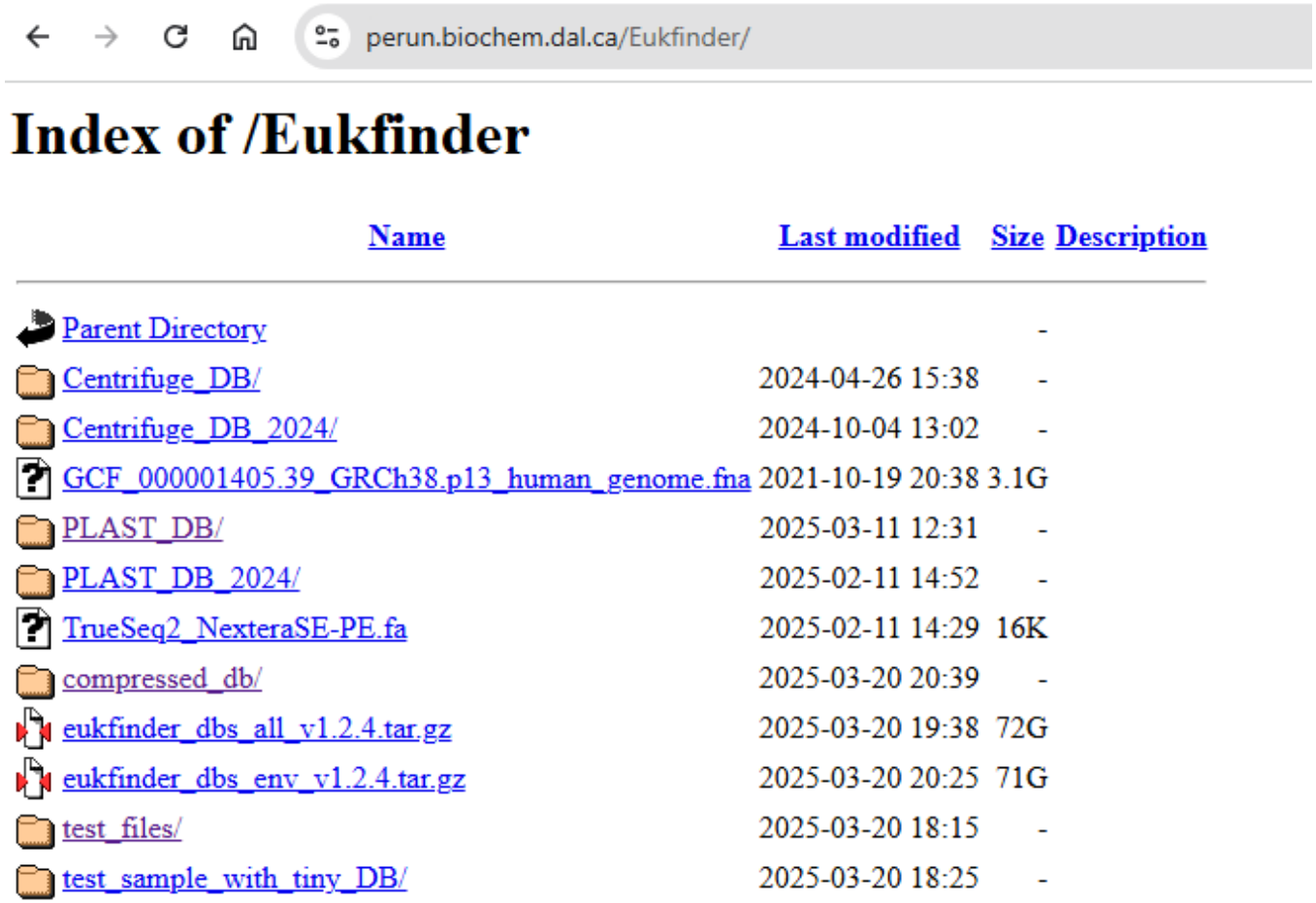
test datasets (long & short) with output

test dataset (contigs) with Tiny DBs

**If you don't want to download large databases to run a test to verify installation, you can just download 6**

# Default reference databases can be downloaded from here:

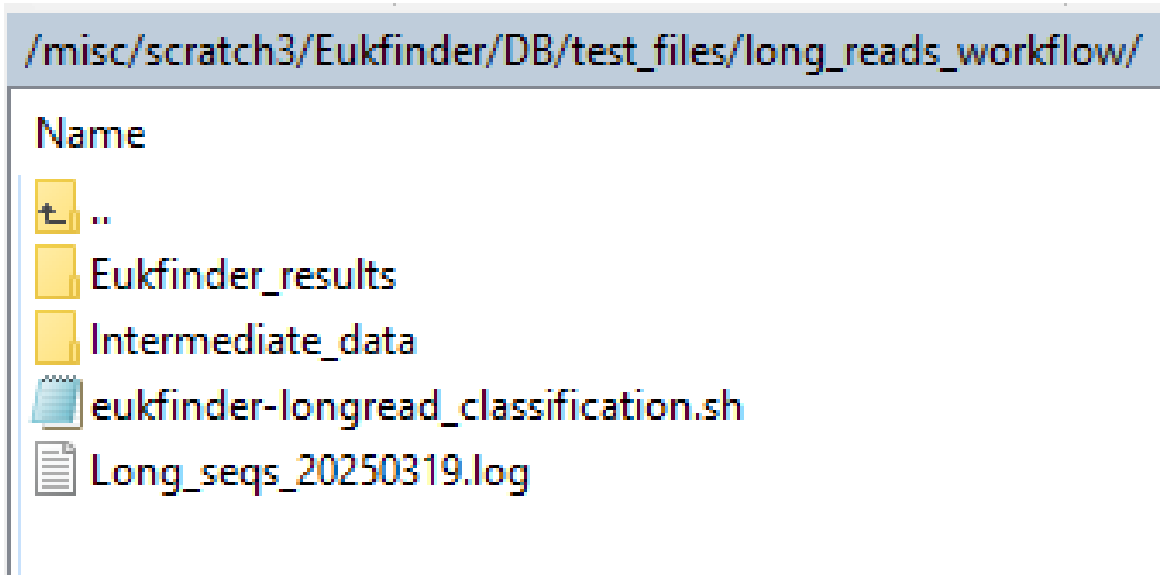
- https://perun.biochem.dal.ca/Eukfinder/



# What to expect from Eukfinder

## long_reads_workflow

Input: fasta / fastq files
(no compressed .gz files)




- Final Result Directory:
  Eukfinder_results
- Log file:
  Long_seqs_20250319.log
- Intermediate File Directory:
  Intermediate_data
  Output from centrifuge/Plast, parsed reads, etc

# What to expect from Eukfinder

## long_reads_workflow

/misc/scratch3/Eukfinder/DB/test_files/long_reads_workflow/

Name

- ..
- Eukfinder_results
- Intermediate_data
- eukfinder-longread_classification.sh
- Long_seqs_20250319.log

- summary_table.txt
- longreads_test.Unk.un.fq
- longreads_test.EUnk.un.fq
- longreads_test.Euk.un.fq
- longreads_test.Bact.un.fq

| Group | #Seq | Total size(bp) |
|---|---|---|
| Unk | 3 | 1420 |
| Euk | 17 | 50839 |
| Bact | 11 | 6429 |
| EUnk | 20 | 52259 |

# How to run Eukfinder on Perun

- **1. Activate Eukfinder environment**

  source activate  eukfinder

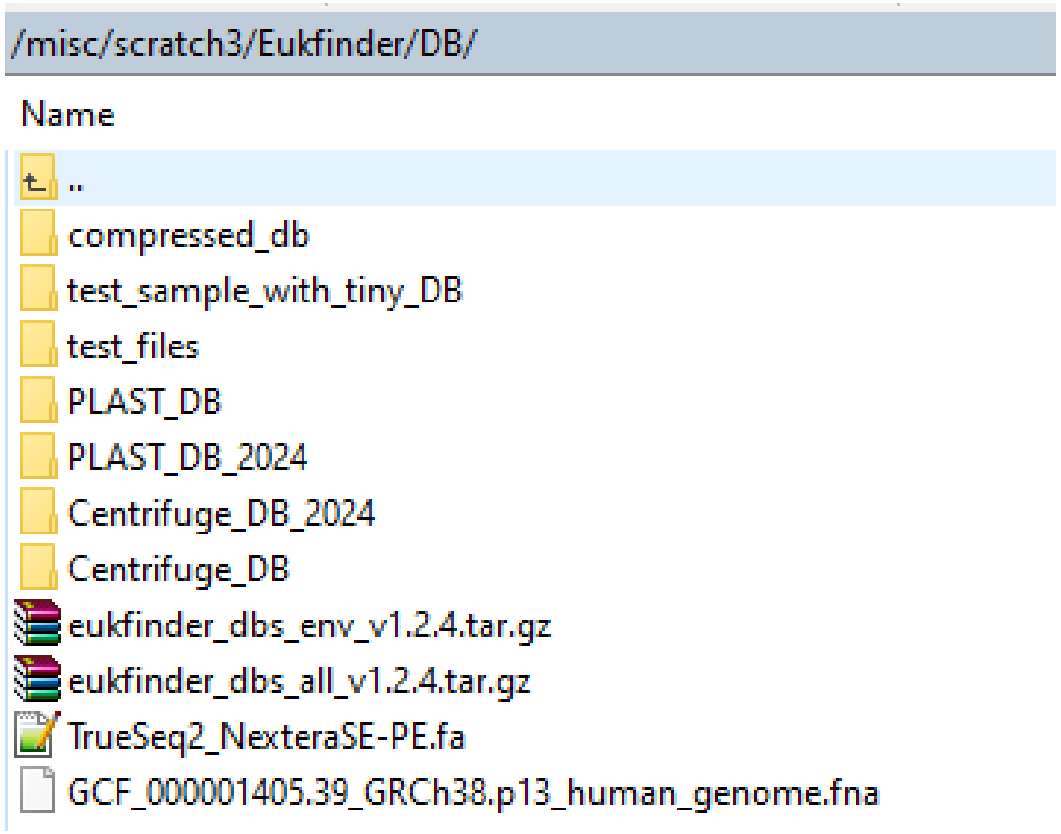
```
(base) dzhao@perun15:~$ source activate eukfinder
(eukfinder) dzhao@perun15:~$
(eukfinder) dzhao@perun15:~$ eukfinder -h
usage: eukfinder [-h] {short_seqs,read_prep,long_seqs} ...

positional arguments:
  {short_seqs,read_prep,long_seqs}

optional arguments:
  -h, --help            show this help message and exit
```

# How to run Eukfinder on Perun

- **2. Database locations**

  **/scratch3/Eukfinder/DB**

  - Centrifuge Database
  - Plast Database
  - Human Genome for read decontamination
  - Read Adapters for Illumina sequencing

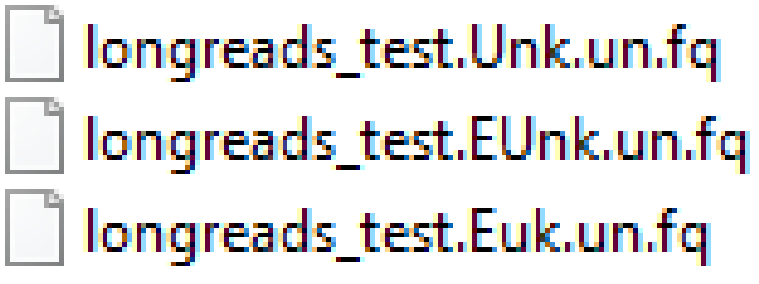


# Running Eukfinder with long reads/assembled contigs

```
(eukfinder) dzhao@perun15:~$ eukfinder long_seqs -h
usage: eukfinder long_seqs [-h] -l LONG_SEQS -o OUT_NAME --mhlen MHLEN --cdb
                           CDB -n NUMBER_OF_THREADS -z NUMBER_OF_CHUNKS -t
                           TAXONOMY_UPDATE -p PLAST_DATABASE -m PLAST_ID_MAP
                           -a ACC2TAX_DATABASE -e E_VALUE --pid PID --cov COV

optional arguments:
  -h, --help            show this help message and exit

Required arguments:
  Description

  -l LONG_SEQS, --long-seqs LONG_SEQS
                        long sequences file
  -o OUT_NAME, --out_name OUT_NAME
                        out name

  -n NUMBER_OF_THREADS, --number-of-threads NUMBER_OF_THREADS
                        Number of threads
  -z NUMBER_OF_CHUNKS, --number-of-chunks NUMBER_OF_CHUNKS
                        Number of chunks to split a file
```

number of chunks the input files is split into for processing with PLAST

-o longreads_test



# Break down of Required arguments

-t, --taxonomy-update  TAXONOMY_UPDATE

- set to **True** the first time you run or didn't run it for a while
- otherwise set to False
- Use python package ete3 to convert taxoID to taxonomy
- download taxonomy file to your personal folder

```
# --- dumping taxonomy if requested ---
ncbi = ete3.NCBITaxa()
if tax_up:
    ncbi.update_taxonomy_database()
```

# Break down of Required arguments -- path to databases (Must be full path/absolute path)

-p PLAST_DATABASE, --plast-database PLAST_DATABASE

path to plast database

-m PLAST_ID_MAP, --plast-id-map PLAST_ID_MAP

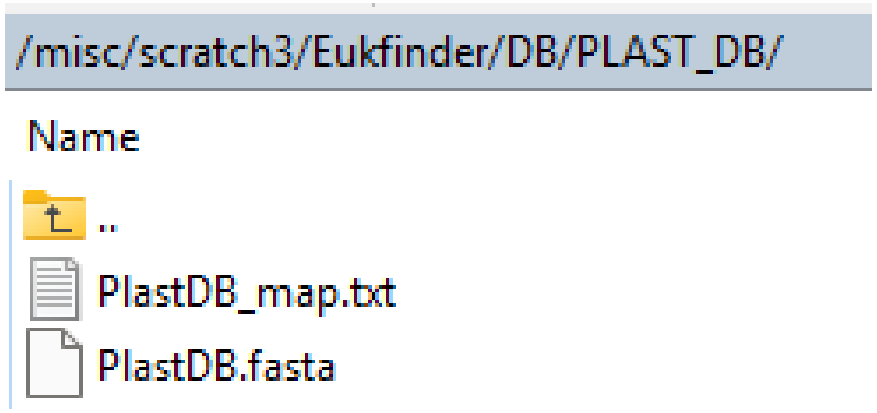
path to taxonomy map for plast database

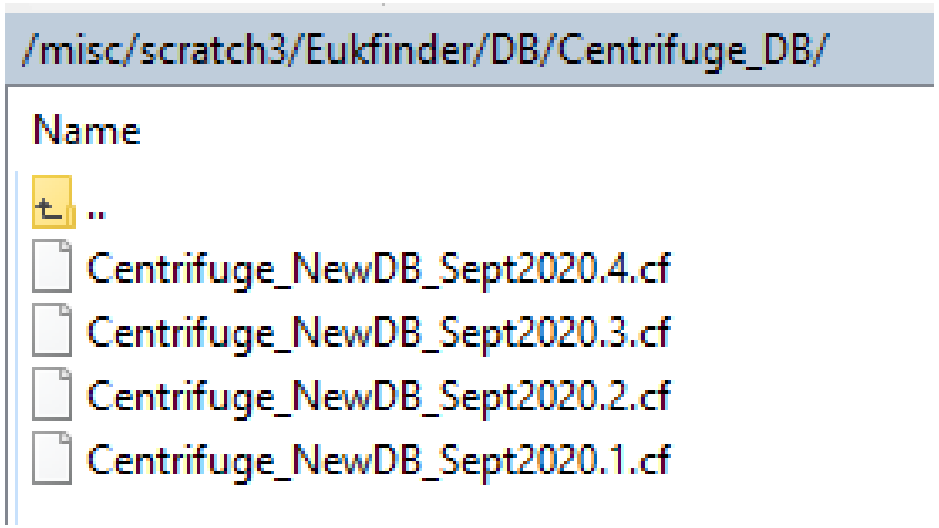
--cdb CDB, --centrifuge-database CDB

Basename of Centrifuge index files (exclude .1.cf suffix).

e.g., for files Centrifuge_NewDB_Sept2020.1.cf, .2.cf, set:

--cdb /misc/scratch3/Eukfinder/DB/Centrifuge_DB/Centrifuge_NewDB_Sept2020



# Break down of Required arguments -- parameters for classifiers

-e E_VALUE, --e-value E_VALUE (PLAST)

threshold for plast searches

--pid PID, --percent_id PID (PLAST)

percentage identity for plast searches

--cov COV, --coverage COV (PLAST)

percentage coverage for plast searches

--mhlen MHLEN, --min-hit-length MHLEN (Centrifuge)

minimum hit length

| Value | Lenient | Strict |
|---|---|---|
| -e E_VALUE | 0.01 | 1e-5 |
| --pid PID | 60 | 80 |
| --cov COV | 15/20 | 30 |
| --mhlen MHLEN | Short: 30<br>Long: 50 | Short: 40<br>Long: 100 |

https://github.com/RogerLab/Eukfinder/blob/main/command_lines_by_task/eukfinder_long_classify.sh

```
# 1. Define input files and output prefix
input="read_fastq_or_assembly_fasta"
prefix="Test_long"

# 2. Full paths to databases (edit these as needed)
plastdb="/path/to/Plast_DB.fasta"       # Path to the PLAST database
plastmap="/path/to/Plast_DB.map"        # Path to the PLAST map file
centrifuge="/path/to/Centrifuge_DB"     # base name (no .1.cf, .2.cf, etc.)
acc2tax="/path/to/Acc2Tax_DB"           # Path to folder of acc2tax database

# 3. Activate Conda environment
source "$(conda info --base)/etc/profile.d/conda.sh"
conda activate Eukfinder
```

```
# 4. Run Eukfinder for long reads
#    - See `eukfinder long_seqs -h` for additional flags.
eukfinder long_seqs \
    -l "$input" \
    -n 18 \
    -z 3 \
    -t False \
    -p "$plastdb" \
    -m "$plastmap" \
    -a "$acc2tax" \
    -e 0.01 \
    --pid 80 \
    --cov 30 \
    -o "$prefix" \
    --cdb "$centrifuge" \
    --mhlen 40

# 5. Deactivate Conda environment
conda deactivate
```

# Running Eukfinder with short read Illumina data

```
(eukfinder) dzhao@perun15:~$ python3 /scratch3/Eukfinder/eukfinder.py short_seqs -h
usage: eukfinder short_seqs [-h] --r1 R1 --r2 R2 --un UN -o OUT_NAME
                            [-n NUMBER_OF_THREADS] [-z NUMBER_OF_CHUNKS]
                            [-t TAXONOMY_UPDATE] [-p PLAST_DATABASE]
                            [-m PLAST_ID_MAP] [--cdb CDB] [-e E_VALUE]
                            [--pid PID] [--cov COV] [--max_m MAX_M] [-k KMERS]
                            [--mhlen MHLEN] [--pclass PCLASS]
                            [--uclass UCLASS]
```

## Parameters specific to Eukfinder_short

```
# Full path to previously prepared reads

      --r1 R1, --reads-r1 R1
                            left reads
      --r2 R2, --reads-r2 R2
                            right reads
      --un UN, --un-pair-reads UN
                            orphan reads
```

Input: fastq files
(no compressed .gz files)

# Running Eukfinder with short read Illumina data

## Parameters specific to Eukfinder_short

- Two rounds of classification by Centrifuge:

  1st: run in **read_prep** step

  Given location of centrifuge results by using arguments --pclass , --uclass

```
--pclass PCLASS, --p-reads-class PCLASS
                      Classification for pair end reads
--uclass UCLASS, --u-reads-class UCLASS
                      Classification for un-pair end reads
```

  2nd: run in **short_seqs** step

  Define centrifuge database location and min-hit-length here

```
--mhlen MHLEN, --min-hit-length MHLEN
                        minimum hit length
```

# Running Eukfinder with short read Illumina data

## Parameters specific to Eukfinder_short

- Assemble classified reads using metaSPAdes:

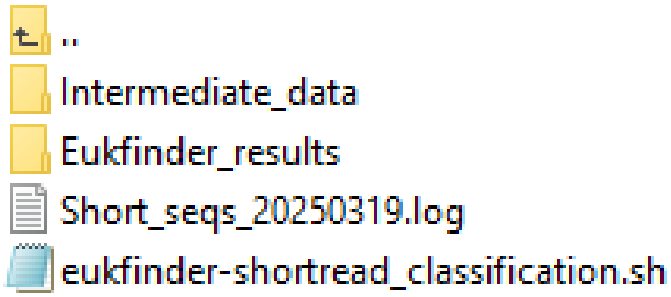
```
--max_m MAX_M, --max_memory MAX_M
                      Maximum memory allocated to carry out an assembly
-k KMERS, --kmers KMERS
                      kmers to use during assembly. These must be odd and
                      less than 128. default is 21,33,55
```

# What to expect from Eukfinder_short

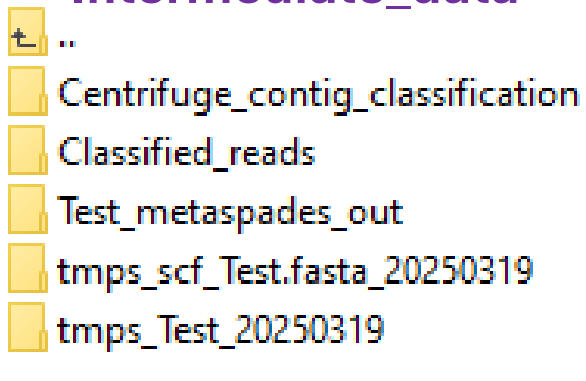
## short_reads_workflow

/misc/scratch3/Eukfinder/DB/test_files/short_reads_workflow/

Name




**Intermediate_data**




- 1. tmps_OUTPUTNAME_DATE
  Intermediate files from 1st round of classification
- 2. Classified_reads
  short reads classified as bacterial, eukaryote, archaea, misc or unk
- 3. OUTPUTNAME _metaspades_out
  assemble sequences indentified as Eukaryote or unknown using metaSdes
- 4. Centrifuge_contig_classification
  output from metaSdes and 2nd round of centrifuge classification
- 5. tmps_scf_OUTPUTNAME.fasta_DATE
  Intermediate files from 2nd round of classification

# Eukfinder short read preparation

1. **Remove adapters, low quality data, and host sequences**

   Trimmomatic

2. **Remove host sequences (only in read_prep mode)**

   Bowtie2

3. **Run centrifuge for 1st round of classification**

# Running Eukfinder short read preparation

**Two modes:**

one for samples need to remove host genome: read_prep

one for **environment samples** without host genome: read_prep_env

```
(eukfinder) dzhao@perun15:~$ python3 /scratch3/Eukfinder/eukfinder.py read_prep -h
usage: eukfinder read_prep [-h] --r1 R1 --r2 R2 -n THREADS -i ILLUMINA_CLIP
                           --hcrop HCROP -l LEADING_TRIM -t TRAIL_TRIM --wsize
                           WSIZE --qscore QSCORE --mlen MLEN --mhlen MHLEN
                           --hg HG -o OUT_NAME [--cdb CDB] [--qenc QENC]
```

```
(eukfinder) dzhao@perun15:~$ python3 /scratch3/Eukfinder/eukfinder.py read_prep_env -h
usage: eukfinder read_prep_env [-h] --r1 R1 --r2 R2 -n THREADS -i
                               ILLUMINA_CLIP --hcrop HCROP -l LEADING_TRIM -t
                               TRAIL_TRIM --wsize WSIZE --qscore QSCORE --mlen
                               MLEN --mhlen MHLEN -o OUT_NAME --cdb CDB
                               [--qenc QENC]
```

Example host genomes: human for gut metagenomes; bacterial genomes for eukaryotic culture sequencing data

# Running Eukfinder short read preparation

## Parameters for Trimmomatic

```
-i ILLUMINA_CLIP, --illumina-clip ILLUMINA_CLIP
                      adaptor file
--hcrop HCROP, --head-crop HCROP
                      head trim
-l LEADING_TRIM, --leading-trim LEADING_TRIM
                      leading trim
-t TRAIL_TRIM, --trail-trim TRAIL_TRIM
                      trail trim
--wsize WSIZE, --window-size WSIZE
                      sliding window size
--qscore QSCORE, --quality-score QSCORE
                      quality score for trimming
--mlen MLEN, --min-length MLEN
                      minimum length
```

```
# path to databases
centrifuge=/Full/path/to/Centrifuge_DB
adapters=/Full/path/to/TrueSeq2_NexteraSE-PE.fa
host_genome=/Full/path/to/test.host.fasta

# command line for read preparation
conda activate eukfinder
python3 eukfinder read_prep --r1 $R1 --r2 $R2 \
                            -n 42 \
                            --hcrop 5 \
                            -l 10 -t 10 \
                            --wsize 40 --qscore 25 \
                            --mlen 30 --mhlen 30 \
                            --hg $host_genome \
                            -i $adapters \
                            --cdb $centrifuge  -o read_prep
```

# Running Eukfinder short read preparation

## Parameters for Trimmomatic

```
--qenc QENC, --quality-encoding QENC
                      quality enconding for trimmomatic
```

required=False

Define this parameter if read_prep or read_prep_env failed
This may happen when trimmomatic can not detect quality enconding
**Purpose**: Specify quality encoding for Trimmomatic manually if automatic detection fails.
**Accepted values**: phred33, phred64
**Default**: Empty (automatic detection by Trimmomatic)
Typical error handled:Error: Unable to detect quality encoding
Example usage: --qenc phred33

# Build custom databases

- Detailed Instruction available:

  https://github.com/RogerLab/Eukfinder/tree/main/Building_custom_DB

- Centrifuge

```
centrifuge-build -p 4 --conversion-table seqid2taxid.map \
                 --taxonomy-tree taxonomy/nodes.dmp --name-table taxonomy/names.dmp \
                 input-sequences.fna abv
```

Centrifuge map file

```
NZ_CP017803.1    2173
NZ_AP025586.1    2173
NZ_AP025587.1    2173
NZ_CP017921.1    2177
NZ_CP113361.1    2199
NZ_CP077107.1    2203
NZ CP042908.1    2209
```

- PLAST

  A sequence fasta & map file

PLAST doesn't support new v5 NCBI nr and nt databases and can cause Segmentation fault error.

PLAST map file

```
CDFH01000266.1  65658    Eukaryota
CDFH01002235.1  65658    Eukaryota
CDFH01002688.1  65658    Eukaryota
CDFH01003784.1  65658    Eukaryota
CDFH01007592.1  65658    Eukaryota
CDFH01007660.1  65658    Eukaryota
CDFH01017633.1  65658    Eukaryota
```

# Build Centrifuge custom database

- **Method 1: Downloading Genomes Directly with centrifuge-download**

```
source activate eukfinder

# Download all complete archaeal, bacterial, and viral genomes and mask low-complexity regions
centrifuge-download -o library -m -d "archaea,bacteria,viral" refseq > seqid2taxid.map

# Merge all downloaded sequences into one file
cat library/*/*.fna > input-sequences.fna

# Download NCBI taxonomy files
centrifuge-download -o taxonomy taxonomy

# Build the Centrifuge index
centrifuge-build -p 20 --conversion-table seqid2taxid.map \
                 --taxonomy-tree taxonomy/nodes.dmp --name-table taxonomy/names.dmp \
                 input-sequences.fna abv
```

# centrifuge-download arguments

- Use with caution if download a large amount of genomes

- `seqid2taxid.map` may miss taxoID for some of the sequences

```
centrifuge-download [<options>] <database>

ARGUMENT
 <database>        One of refseq, genbank, contaminants or taxonomy:
                     - use refseq or genbank for genomic sequences,
                     - contaminants gets contaminant sequences from UniVec and EmVec,
                     - taxonomy for taxonomy mappings.

COMMON OPTIONS
 -o <directory>         Folder to which the files are downloaded. Default: '.'.
 -P <# of threads>      Number of processes when downloading (uses xargs). Default: '1'

WHEN USING database refseq OR genbank:
 -d <domain>            What domain to download. One or more of bacteria, viral, archaea, fungi,
 protozoa, invertebrate, plant, vertebrate_mammalian, vertebrate_other (comma separated).
 -a <assembly level>    Only download genomes with the specified assembly level. Default: 'Compl
ete Genome'. Use 'Any' for any assembly level.
 -c <refseq category>   Only download genomes in the specified refseq category. Default: any.
 -t <taxids>            Only download the specified taxonomy IDs, comma separated. Default: any.
 -g <program>           Download using program. Options: rsync, curl, wget. Default curl (auto-d
etected).
 -r                     Download RNA sequences, too.
 -u                     Filter unplaced sequences.
 -m                     Mask low-complexity regions using dustmasker. Default: off.
 -l                     Modify header to include taxonomy ID. Default: off.
 -v                     Verbose mode
```

# Build Centrifuge custom database

- **Method 2: Using ncbi-genome-download**
- Download the desired genomes by specifying a genus or taxonomic group

  source activate ncbi-genome-download

  ncbi-genome-download --genera Blastocystis -p 4 -r 10 --flat-output --progress-bar --formats fasta,assembly-report protozoa

- Combine all downloaded FASTA files into a single file

  cat *.fna > genome.fasta

- Generate the sequence-to-taxid mapping file using a custom script

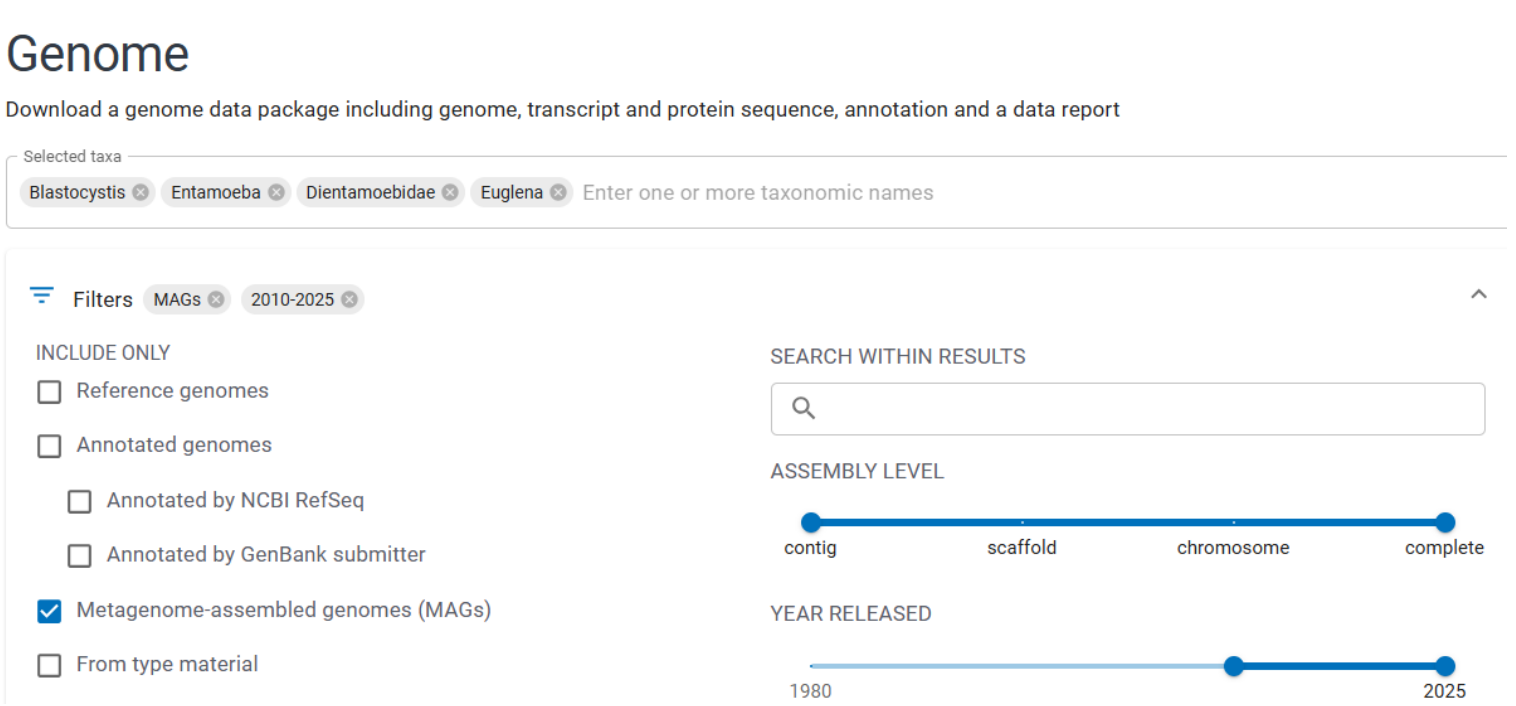
  Example script available on Github:

  Build_Centrifuge_map_from_assembly_report.py

# Build Centrifuge custom database

- **Method 3: Manually select Genome Accession Numbers**

1. Choose "Genome" as search criteria, enter taxonomy name
2. Filter results (reference genomes, MAGs, assembly level, release date)
3. Download the table and extract accession numbers from the filtered results




66 Genomes

| GenBank | RefSeq | Scientific name |
|---|---|---|
| GCA_948471165.1 | | Blastocystis hominis |
| GCA_937873115.1 | | Blastocystis sp. subtype 4 |
| GCA_937862745.1 | | Blastocystis sp. subtype 4 |
| GCA_029241775.1 | | Blastocystis sp. subtype 3 |
| GCA_025449415.1 | | Blastocystis sp. subtype 3 |
| GCA_029241725.1 | | Blastocystis sp. subtype 3 |
| GCA_029241515.1 | | Blastocystis sp. subtype 3 |
| GCA_029241095.1 | | Blastocystis sp. subtype 3 |
| GCA_029241145.1 | | Blastocystis sp. subtype 3 |

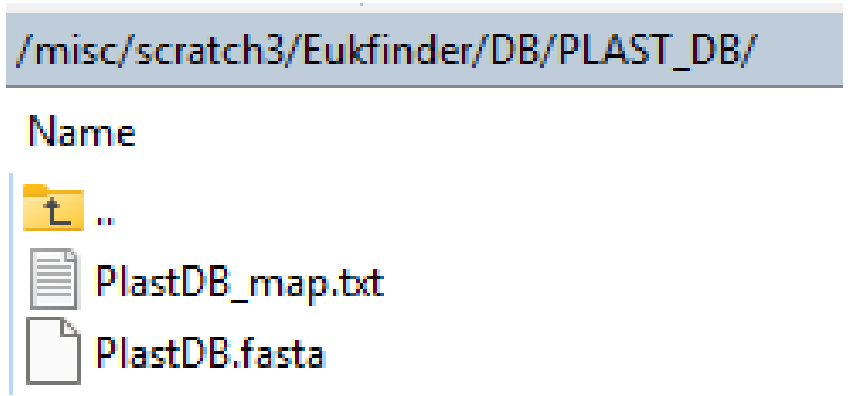
# Build PLAST custom database

- **Preliminary Analysis**
  Use search tools such as DIAMOND on a subset of your data to identify organisms that may be present. This helps you determine which genomes or taxa to include.

- **Focused Selection**
  Choose genomes matching your organisms of interest. Follow a process similar to Method 3 for building a Centrifuge database (i.e., selecting accession numbers, downloading relevant assembly summaries, and retrieving corresponding sequences).

- **Keep the Database Manageable**
  Plast searches can be time-consuming, and large databases significantly increase runtime. Aim for a database ≤30 GB to maintain reasonable search speeds within the Eukfinder workflow

- Combine all downloaded FASTA files into a single fasta
- Generate sequence-to-taxid-group mapping file using in-house script



# Eukfinder Time usage

## Eukfinder_Short on Perun

| # Thread | Memory | Total Time |
|---|---|---|
| 40 | 2T | 3h 5m |
| 30 | 2T | 3h 5m |
| 30 | 256G | 4h 2m |

- Human gut metagenome sample tested on Perun
- Total reads: 17 M

## Eukfinder & similar tools running on Compute Canada Server

| Method | Total Time | Assembling by metaSPAdes | Read/Contigs Classificationa |
|---|---|---|---|
| Eukfinder_Short | 557m 40s | 11m 30s | 546m 10s |
| Eukfinder_Long | 98m 26s | 81m 20s | 17m 6s |
| EukRep | 85m 28s | 81m 20s | 4m 8s |
| Tiara | 82m 28s | 81m 20s | 1m 8s |

- Human gut metagenome sample (threads = 10, except EukRep only use 1 thread)
- Total reads: 30.2 M
- Assembly size: 141 Mb
- Total contigs: 30086 (length ≥1000bp).
- Lenovo SD350 server with 40 Intel “Skylake” cores 2.4 GHz and 188 GB RAM

# Eukfinder Time usage

| Method | Total Time | Assembling by metaSPAdes | Read/Contigs Classificationa |
|---|---|---|---|
| Eukfinder_Short | > 72h (failed) | 37h14m | -- |
| Eukfinder_Long | 48h16m | 47h54m | 22m23s |
| EukRep | 48h | 47h54m | 6m21s |
| Tiara | 47h56m | 47h54m | 1m23s |

- Tara ocean sample (threads = 40, except EukRep only use 1 thread)
- Total reads: 179.4 M
- Assembly size: 148 Mb (47 hours 54 minutes)
- Total contigs: 64,536 (length ≥1000bp).
- Lenovo SD350 server with 40 Intel “Skylake” cores 2.4 GHz and 188 GB RAM

# Database building time usage

| Original fasta size | Database size | Total Time | CPUs | Memory |
|---|---|---|---|---|
| 160 GB | 92 GB | 2 days, 1 hour, 50 minutes | 40 | 768G |
| 283 GB | 175 GB | 3 days, 22 hours, 40 minutes | 80 | 2T |

- Only include times for running `centrifuge-build`

# Strategy for large datasets

**For samples if Eukfinder_short fail to finish:**

**If it finishes the first round of classification**

- Manually assembly after combining reads from euk and unk
- Use assembled contigs as input for Eukfinder long

# Running Eukfinder short & long for Transcriptome

For short reads:

- Test using rnaspades in the assembly step

For assembled contigs (Trinity assembly)

- Run Eukfinder_long with default setting

Future plan:

Building reduced nr plastx database

# Binning

- Binning is the process of grouping assembled DNA fragments (contigs) into clusters that represent individual genomes.
- In metagenomic or transcriptomic data:
  - Each bin ideally corresponds to one organism (e.g., a bacterium, a protist, a fungus).
  - Binning uses features like sequence composition (e.g., GC content), coverage depth, and sometimes taxonomy to group similar contigs together.
- **Binning Workflow available on GitHub**

  https://github.com/RogerLab/MetagenomeBinningWorkflow
- Recommend MyCC

MyCC (Metagenomic Clustering based on Codon usage) is a genome binning tool that groups contigs based on their sequence composition, specifically k-mer (short sequence) frequencies.

Key Points:

- Unsupervised binning: Automatically clusters contigs without prior taxonomic labels.
- Feature: Uses codon usage bias and tetranucleotide frequency patterns.
- Strength: Performs well on complex metagenomes and helps recover microbial genomes (MAGs).
- Limitation: Works best when contigs are relatively long (>1,000 bp).

# Output from MyCC

- Bins (fasta files)
- Cluster.summary (txt file)

| Cluster | WholeGenom | N50 | NoOfCtg | LongestCtgLe | AvgLenOfCtg | Cogs |
|---|---|---|---|---|---|---|
| Cluster.1.fasta | 255354 | 2987 | 95 | 6058 | 1364 | 0 |
| Cluster.2.fasta | 125977 | 3315 | 48 | 5602 | 1325 | 0 |
| Cluster.3.fasta | 99442 | 764 | 166 | 2004 | 301 | 0 |
| Cluster.4.fasta | 144072 | 3743 | 61 | 6411 | 1242 | 0 |
| Cluster.5.fasta | 94194 | 1597 | 83 | 3949 | 579 | 0 |
| Cluster.6.fasta | 314874 | 3225 | 131 | 7123 | 1212 | 0 |
| Cluster.7.fasta | 43959001 | 2812 | 27184 | 35677 | 808 | 2 |
| Cluster.8.fasta | 115279 | 553 | 248 | 1832 | 233 | 1 |
| Cluster.9.fasta | 201000 | 3978 | 63 | 5537 | 1612 | 0 |
| Cluster.10.fasta | 82674 | 1420 | 69 | 2140 | 602 | 0 |
| Cluster.11.fasta | 177898 | 555 | 365 | 2258 | 243 | 26 |
| Cluster.12.fasta | 421367 | 3477 | 179 | 7173 | 1190 | 0 |
| Cluster.13.fasta | 99999 | 2550 | 38 | 5696 | 1338 | 0 |
| Cluster.14.fasta | 120426 | 1800 | 82 | 4351 | 743 | 0 |
| Cluster.15.fasta | 85731 | 648 | 174 | 1485 | 249 | 0 |
| Cluster.16.fasta | 89229 | 1200 | 95 | 2663 | 475 | 0 |
| Cluster.17.fasta | 215575 | 4441 | 56 | 8906 | 1929 | 0 |

How to run MyCC on Perun

```
export PATH=/opt/perun/myCC/Tools:$PATH
source  /scratch2/software/python2-packages/bin/activate
cat Eukfinder_long_EUnk.depth.txt | cut -f 1,3 > Eukfinder_long_EUnk_depth_for_binning.txt
MyCC.py Eukfinder_long.fasta  -a Eukfinder_long_EUnk_depth_for_binning.txt 4mer
MyCC.py Eukfinder_long.fasta  -a Eukfinder_long_EUnk_depth_for_binning.txt 5mer
MyCC.py Eukfinder_long.fasta  -a Eukfinder_long_EUnk_depth_for_binning.txt 56mer
```

Based on my experience: 5mer works best for eukaryotic samples (less clusters), easy to handle

# Output from MyCC

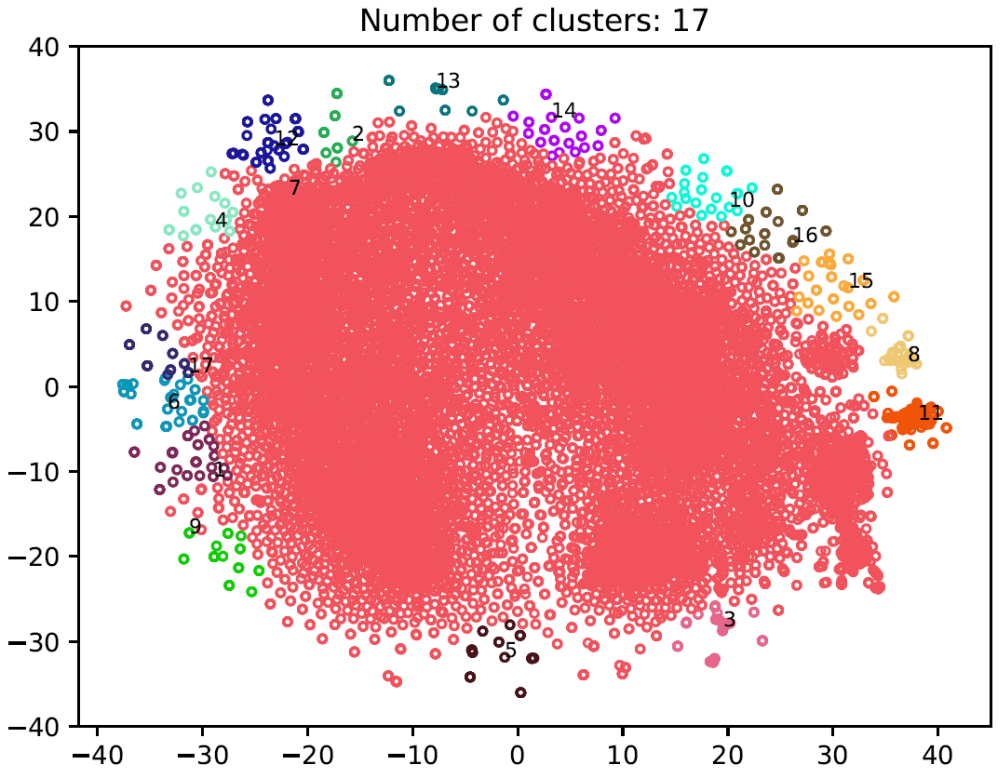
- Bins (fasta files)
- Cluster.summary (txt file)
- MyCluster.pdf

  Corrected clusters based on marker genes
- MyCluster.coords

  Coordinates and cluster number for each contigs

  Notes: overlapping contigs are not shown in this table

| Cluster | WholeGenom | N50 | NoOfCtg | LongestCtgLe | AvgLenOfCtg | Cogs |
|---|---|---|---|---|---|---|
| Cluster.1.fasta | 255354 | 2987 | 95 | 6058 | 1364 | 0 |
| Cluster.2.fasta | 125977 | 3315 | 48 | 5602 | 1325 | 0 |
| Cluster.3.fasta | 99442 | 764 | 166 | 2004 | 301 | 0 |
| Cluster.4.fasta | 144072 | 3743 | 61 | 6411 | 1242 | 0 |
| Cluster.5.fasta | 94194 | 1597 | 83 | 3949 | 579 | 0 |
| Cluster.6.fasta | 314874 | 3225 | 131 | 7123 | 1212 | 0 |
| Cluster.7.fasta | 43959001 | 2812 | 27184 | 35677 | 808 | 2 |
| Cluster.8.fasta | 115279 | 553 | 248 | 1832 | 233 | 1 |
| Cluster.9.fasta | 201000 | 3978 | 63 | 5537 | 1612 | 0 |
| Cluster.10.fasta | 82674 | 1420 | 69 | 2140 | 602 | 0 |
| Cluster.11.fasta | 177898 | 555 | 365 | 2258 | 243 | 26 |
| Cluster.12.fasta | 421367 | 3477 | 179 | 7173 | 1190 | 0 |
| Cluster.13.fasta | 99999 | 2550 | 38 | 5696 | 1338 | 0 |
| Cluster.14.fasta | 120426 | 1800 | 82 | 4351 | 743 | 0 |
| Cluster.15.fasta | 85731 | 648 | 174 | 1485 | 249 | 0 |
| Cluster.16.fasta | 89229 | 1200 | 95 | 2663 | 475 | 0 |
| Cluster.17.fasta | 215575 | 4441 | 56 | 8906 | 1929 | 0 |



# Binning Eukfinder Results Using Anvi'o

- **Why Use Anvi'o for Eukfinder Results?**
  - ✓Visual and interactive genome binning interface
  - ✓Ideal for refining complex metagenomic assemblies
  - ✓Helps separate nuclear, mitochondrial, and contaminant contigs
  - ✓Flexible integration with Eukfinder outputs (e.g., EUnk.fasta)

- **When to Use Anvi’o**
  - ✓Visual control over binning
  - ✓Handling multiple genomes or trying to separate host vs symbiont vs contaminant
  - ✓Have some knowledge (taxonomy, coverage, rRNA) to guide manual refinement
- **Use something else or combine with Anvi’o IF:**
  - ✓Prefer fully automated eukaryote binning (MetaBAT2, MaxBin2, CONCOCT)
  - ✓Data set is massive, and lack of RAM or GPU resources

# Anvi'o

- DokuWiki:

  https://perun.biochem.dal.ca/user-wiki/doku.php?id=decontamination_using_a_metagenomics_approach

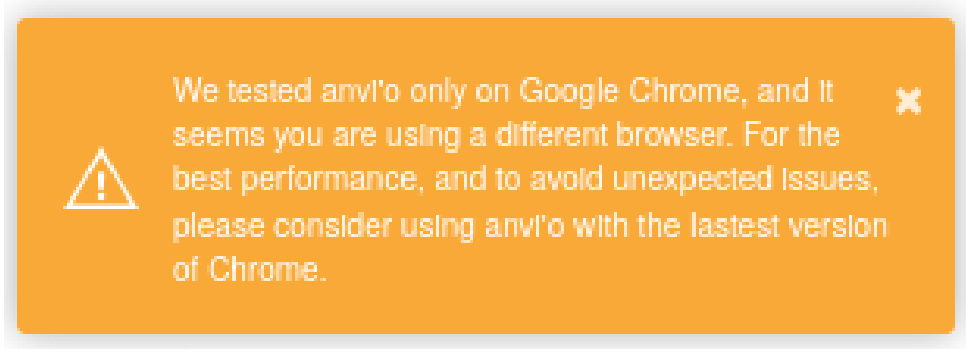
- Running bash available on Perun:

  /scratch3/rogerlab_databases/other_dbs/Anvio7

- Don't run on perun23: open firefox, instead of Chrome
- Interactive may not work on Mac
- Alternative way of running on Perun:

  download profils and config.db to a Linux system

  Newest version v8 (Sept 2023) not installed on Perun



# Overview of the Anvi'o Workflow

1. Prepare contigs
2. Create contigs database
3. Map reads (short & long)
4. Profile mapping
5. Merge profiles
6. Import taxonomy & annotations
7. Visual binning in anvi-interactive

# Running bash shells

## Step 1 - Prep Contigs

```
anvi-script-reformat-fasta input.fasta -o cleaned_contigs.fa -l 1000 --simplify-names
```

Reformat FASTA file (remove contigs based on length, or based on a given list of deflines, and/or generate an output with simpler names)

- -l MIN_LENGTH (Default including all contigs, setting to 1000 in the bash)

## Step 2 - Build Contigs DB

- Generate a new anvi'o contigs database
- Runs all 9 HMM profiles installed on your system
- Get protein sequences for gene calls

```
available ones: "Protista_83" (type: singlecopy),
"Ribosomal_RNA_18S" (type: Ribosomal_RNA_18S),
"Ribosomal_RNA_16S" (type: Ribosomal_RNA_16S),
"Ribosomal_RNA_28S" (type: Ribosomal_RNA_28S),
"Ribosomal_RNA_5S" (type: Ribosomal_RNA_5S),
"Ribosomal_RNA_12S" (type: Ribosomal_RNA_12S),
"Ribosomal_RNA_23S" (type: Ribosomal_RNA_23S),
"Bacteria_71" (type: singlecopy), "Archaea_76" (type
singlecopy). (default: None)
```

```
anvi-gen-contigs-database -f cleaned_contigs.fa -o contigs.db
anvi-run-hmms -c contigs.db -T 40
anvi-get-sequences-for-gene-calls --get-aa-sequences -c contigs.db -o contigs.predicted_orfs
```

# Tips for how to choose minimal contig length

- Generate a series of fasta files with different min lengths:
- i.e., 200, 300, 400, 500, … & All-contigs
- Run BUSCO to check the completion of the genomes

| minimal contig length | BUSCO eukaryota_odb10 |
|---|---|
| All contigs | C:60.0%[S:52.2%,D:7.8%],F:8.6%,M:31.4% |
| 300 | C:60.0%[S:52.2%,D:7.8%],F:8.6%,M:31.4% |
| 400 | C:60.0%[S:52.2%,D:7.8%],F:8.6%,M:31.4% |
| 500 | C:60.0%[S:52.2%,D:7.8%],F:8.6%,M:31.4% |
| 600 | C:59.6%[S:51.8%,D:7.8%],F:8.6%,M:31.8% |
| 700 | C:58.9%[S:51.4%,D:7.5%],F:8.6%,M:32.5% |
| 800 | C:58.1%[S:50.6%,D:7.5%],F:8.6%,M:33.3% |

# Running bash shells

- step3-ShortReads.MAP
- step3-Longreads_MAP

  ✓Mapping reads to fasta to get depth of coverage

- step3-DiamondSearch

  Diamond output needs to be parsed for import taxonomy

  source activate python36-generic

  python python36-acc2taxextraplusv2plast.py contigs.predicted_orfs.tsv

| gene_callers_id | t_domain | t_phylum | t_class | t_order | t_family | t_genus | t_species |
|---|---|---|---|---|---|---|---|
| 1 | Eukaryota | Discoba | Euglenozoa | Kinetoplastea | Metakinetoplastina | Trypanosomatida | Phytomonas sp. EM1 |
| 10 | Bacteria | Pseudomonadota | Gammaproteobacteria | Oceanospirillales | Saccharospirillaceae | Saccharospirillum | Saccharospirillum sp. |
| 1000 | Bacteria | Pseudomonadota | Gammaproteobacteria | Oceanospirillales | Saccharospirillaceae | Saccharospirillum | Saccharospirillum sp. |
| 10001 | Eukaryota | Discoba | Euglenozoa | Kinetoplastea | Metakinetoplastina | Eubodonida | Bodo saltans |
| 10002 | Eukaryota | Discoba | Euglenozoa | Kinetoplastea | Metakinetoplastina | Eubodonida | Bodo saltans |
| 10003 | Eukaryota | Discoba | Euglenozoa | Kinetoplastea | Metakinetoplastina | Eubodonida | Bodo saltans |

# Running bash shells

- step4-ImportTax
  - ✓Import gene-level taxonomy into an anvi'o contigs database
  - ✓Affiliate single-copy core genes in an anvi'o contigs database with taxonomic names
  - ✓--min-percent-identity 50 (original setting is 90, may need to change based on your data)
- step5-MergeProfiles: only for samples with both short and long reads
- step6-ExportSplits-taxonomy

```
c_000000000003_split_00001      Trypanosomatida
c_000000000004_split_00001      Trypanosomatida
c_000000000005_split_00001
```

# Interactive Binning

- MobaXterm: log in to Perun

```
source activate anvio7
```

```
anvi-interactive -c contigs.db -p MERGED_SAMPLES/PROFILE.db --taxonomic-level t_phylum
```

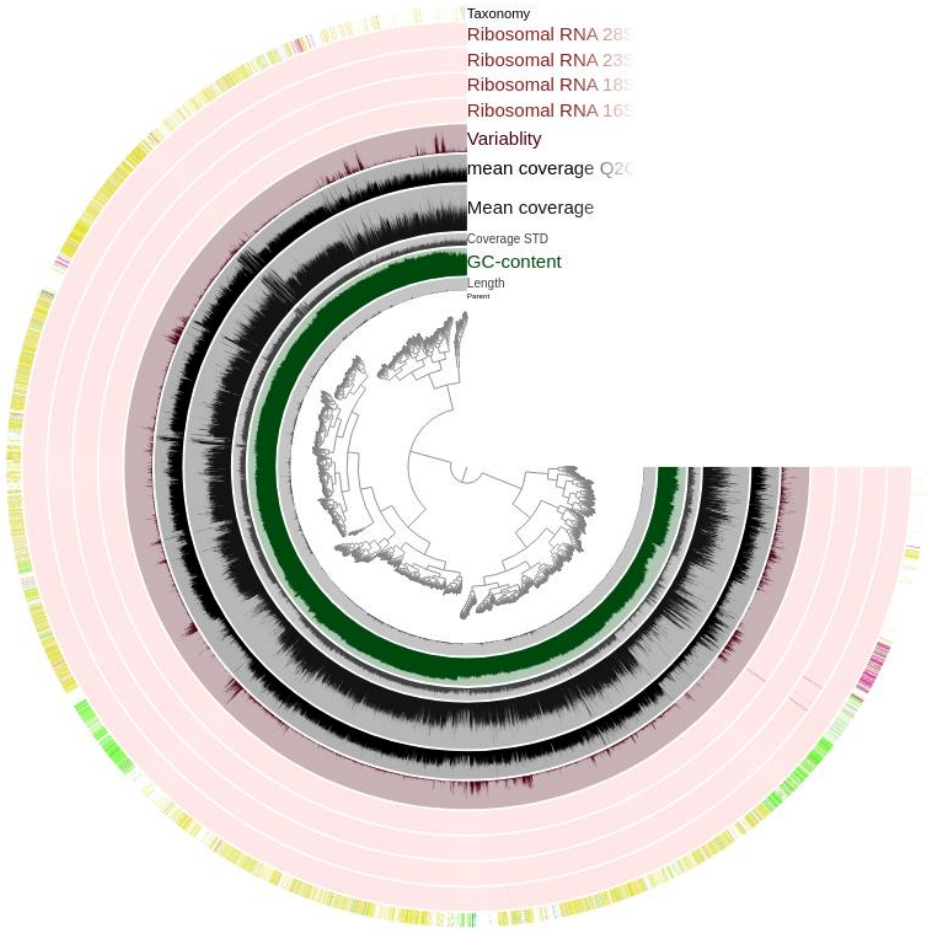
- Press “DRAW” in the browser window that pops up to see the graph

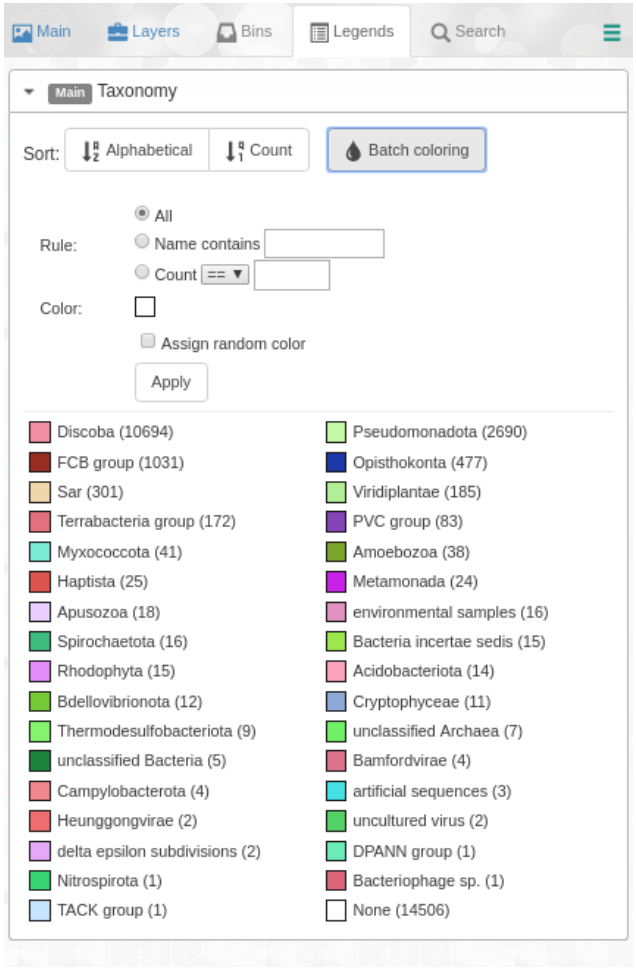
## Not necessary:

- Either use a remote Desktop client software such as X2go to “dig a tunnel” to a perun node
- type in: “ssh -X -Y -L 8080:localhost:8080 youraccount@perun.biochem.dal.ca” in a new terminal

# How to re-color the layers

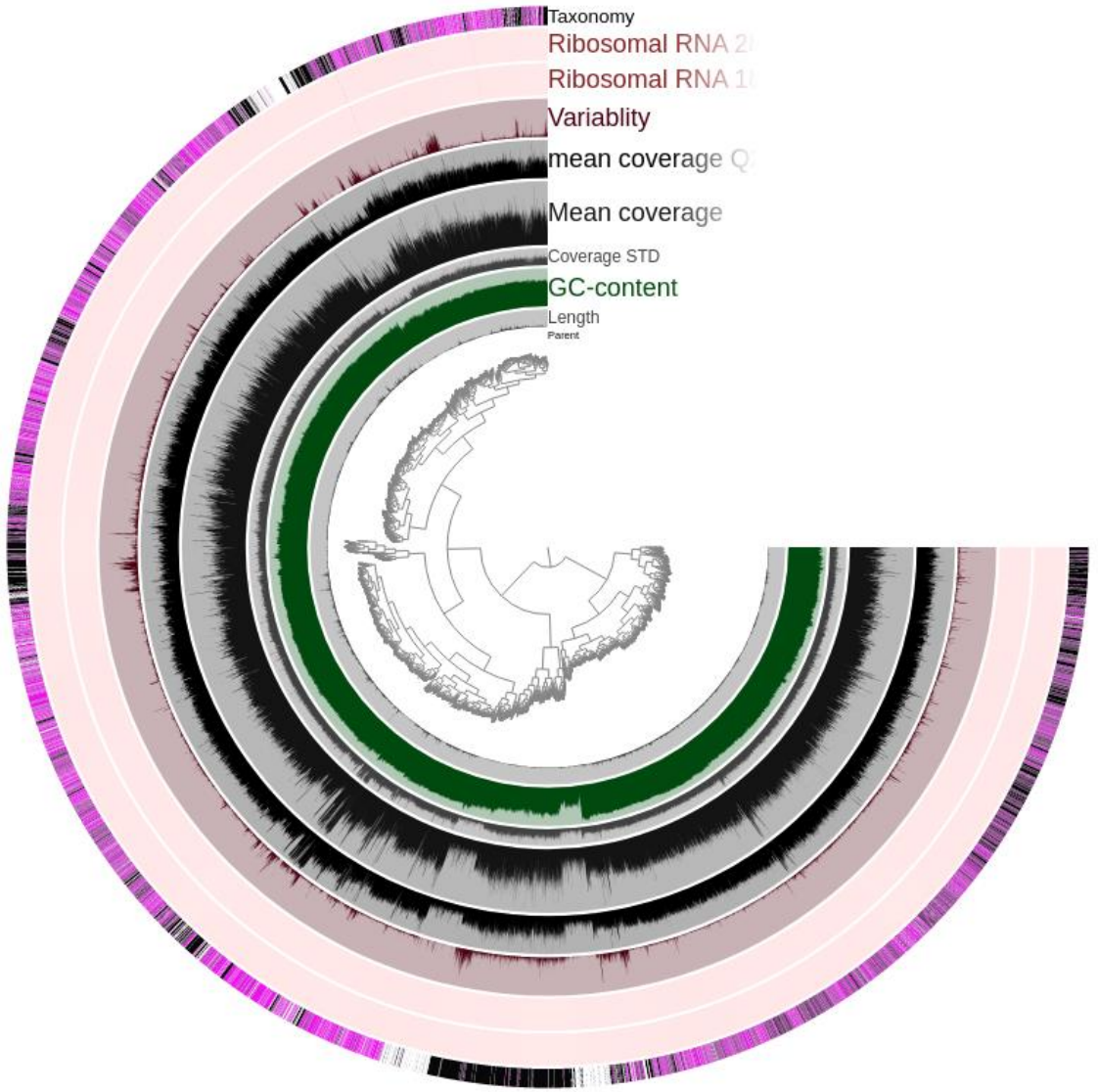
1. In Legends panel: Batch coloring -> Apply: Set all taxonomy to white
2. Set color to the ones you want to see, then click Draw



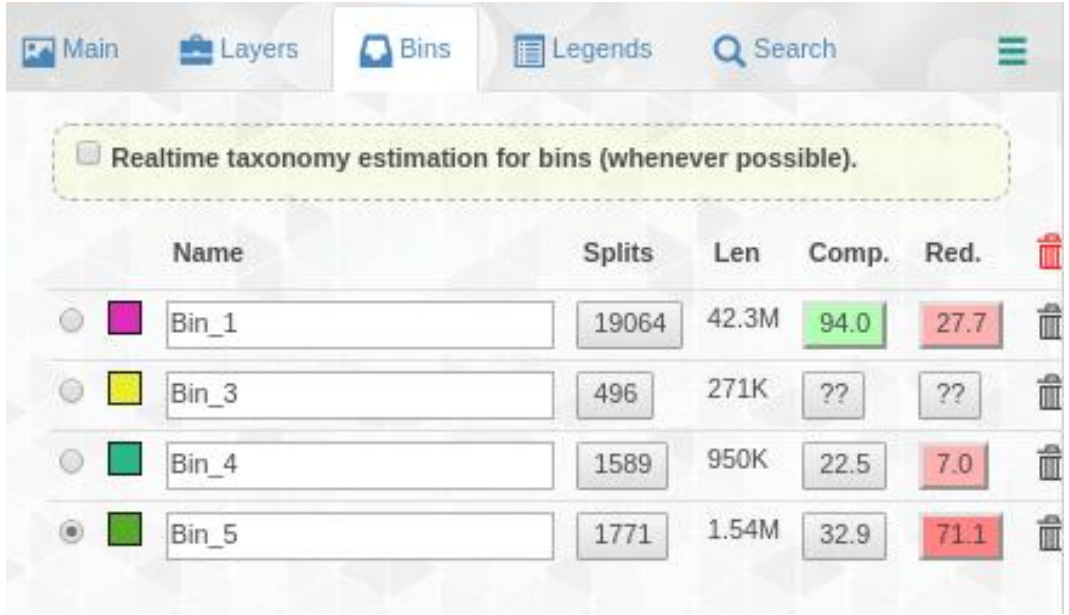
# Re-color the Taxonomy layer

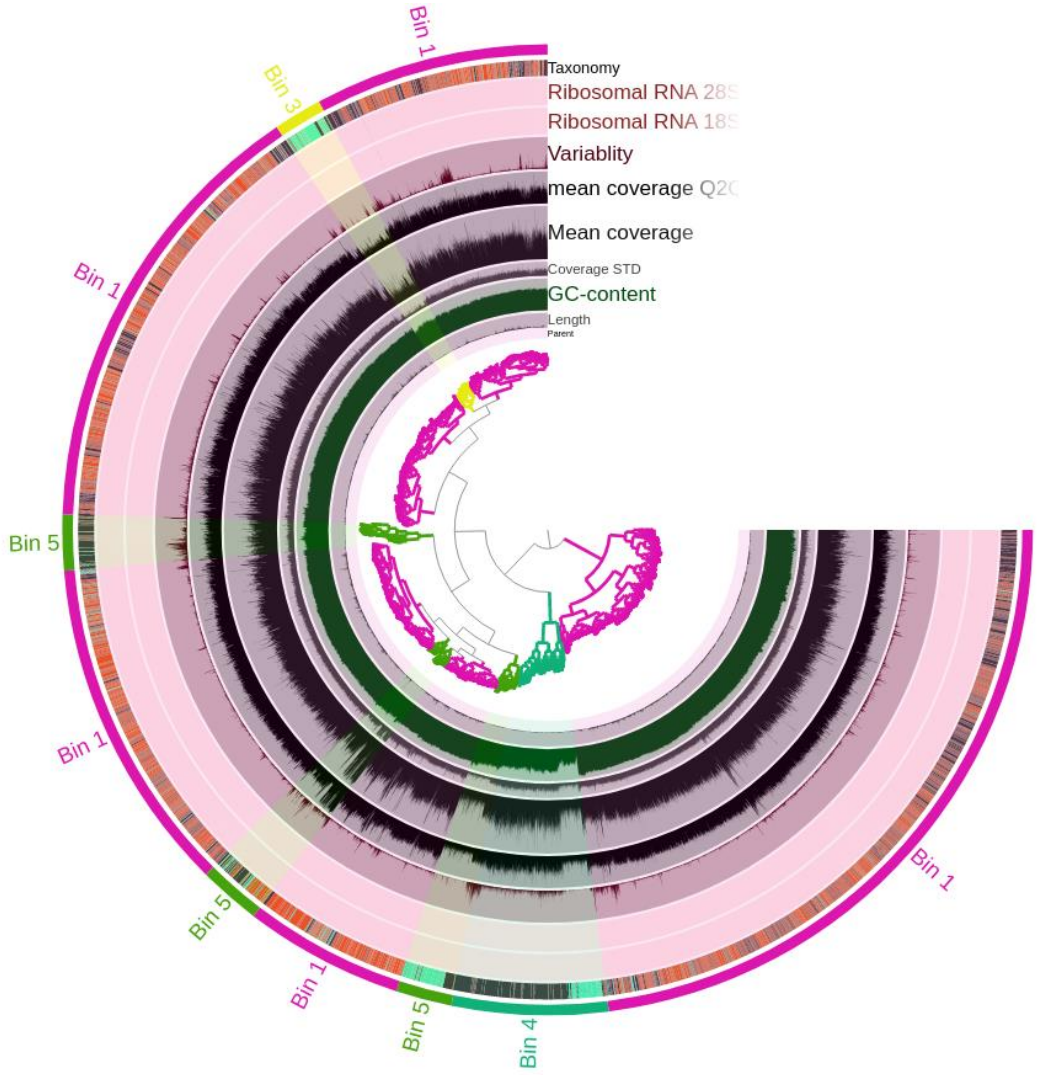
- Discoba (10321)
- FCB group (666)
- Sar (292)
- Pseudomonadati (136)
- PVC group (71)
- Metazoa (37)
- Bacillati (26)
- Metamonada (23)
- Bacteria (17)
- Rhodophyta (16)
- Spirochaetota (14)
- Archaea (9)
- Fungi (5)
- artificial sequences (3)
- virus (2)
- None (9316)
- Pseudomonadota (1047)
- Opisthokonta (436)
- Viridiplantae (178)
- Terrabacteria group (126)
- Amoebozoa (40)
- Myxococcota (37)
- Haptista (23)
- Bacteria incertae sedis (19)
- Acidobacteriota (17)
- Apusozoa (16)
- Cryptophyceae (12)
- Thermodesulfobacteriota (6)
- Viruses (5)
- MAG: adenylate guanylate cyclase domain-containing protein bacterium (3)
- MAG TPA_asm: M14 family zinc carboxypeptidase bacterium (1)



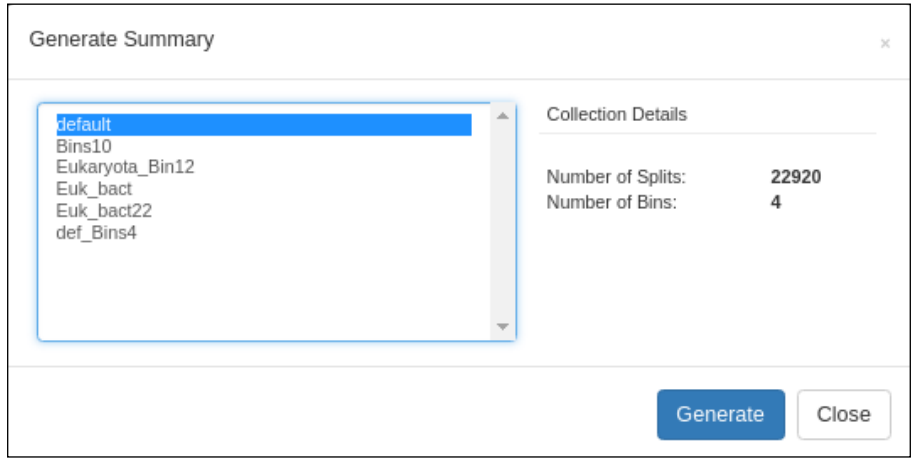
# Manual select bins

- Group contigs with similar profiles manually
- Look at coverage, GC content, taxonomy, and split quality

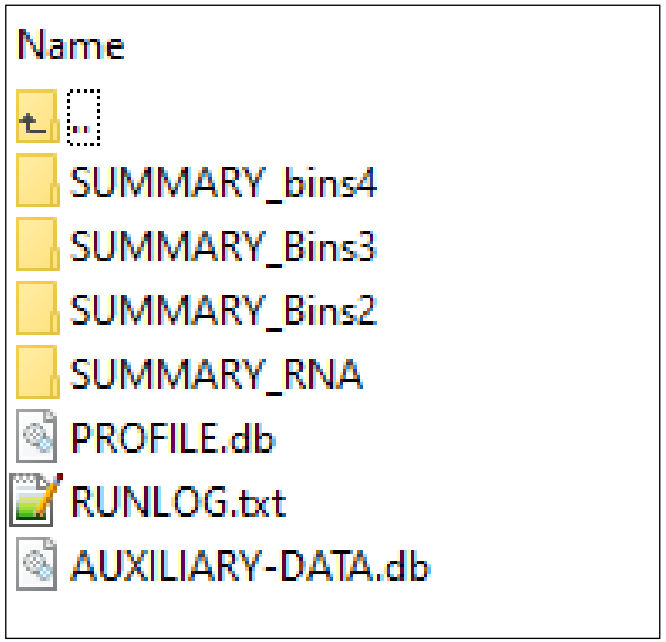


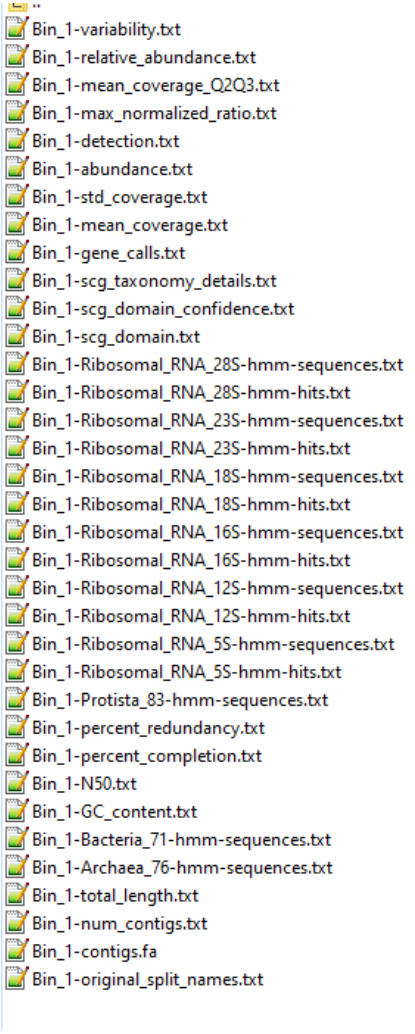



# Store bins

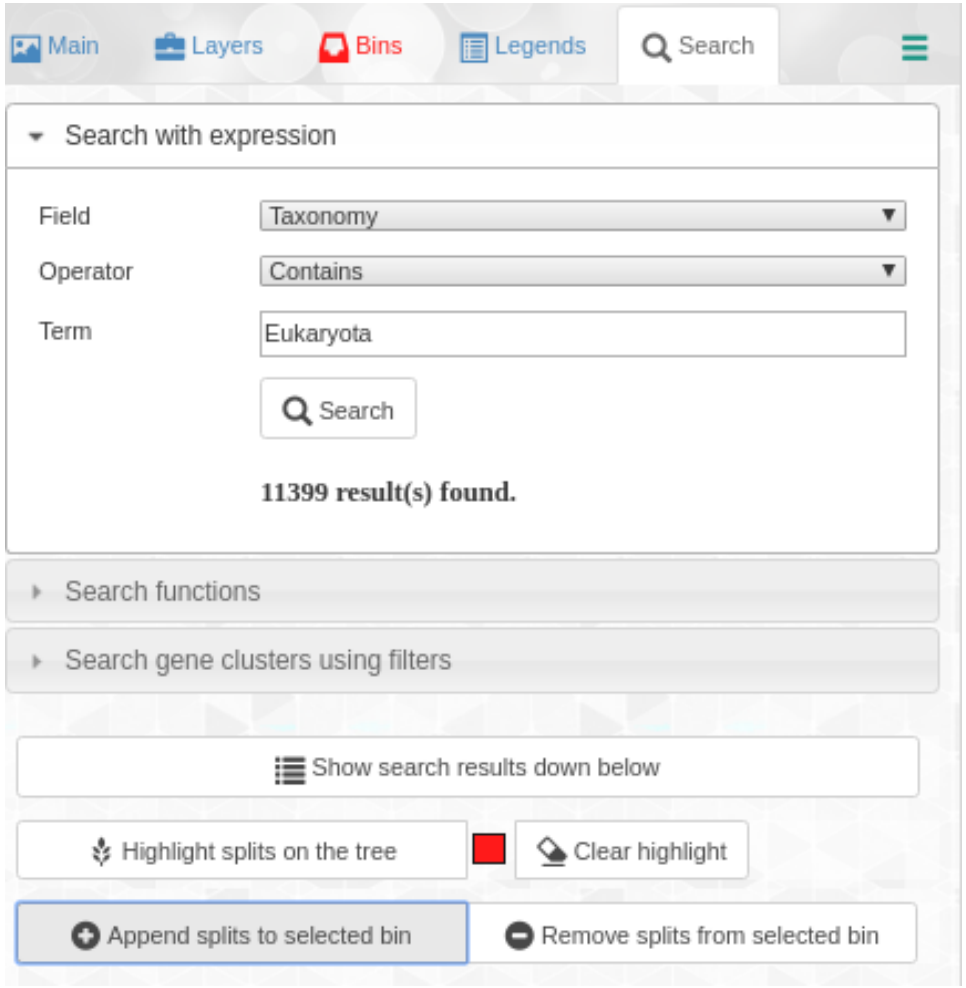

All the bins will be stored in ProfDir

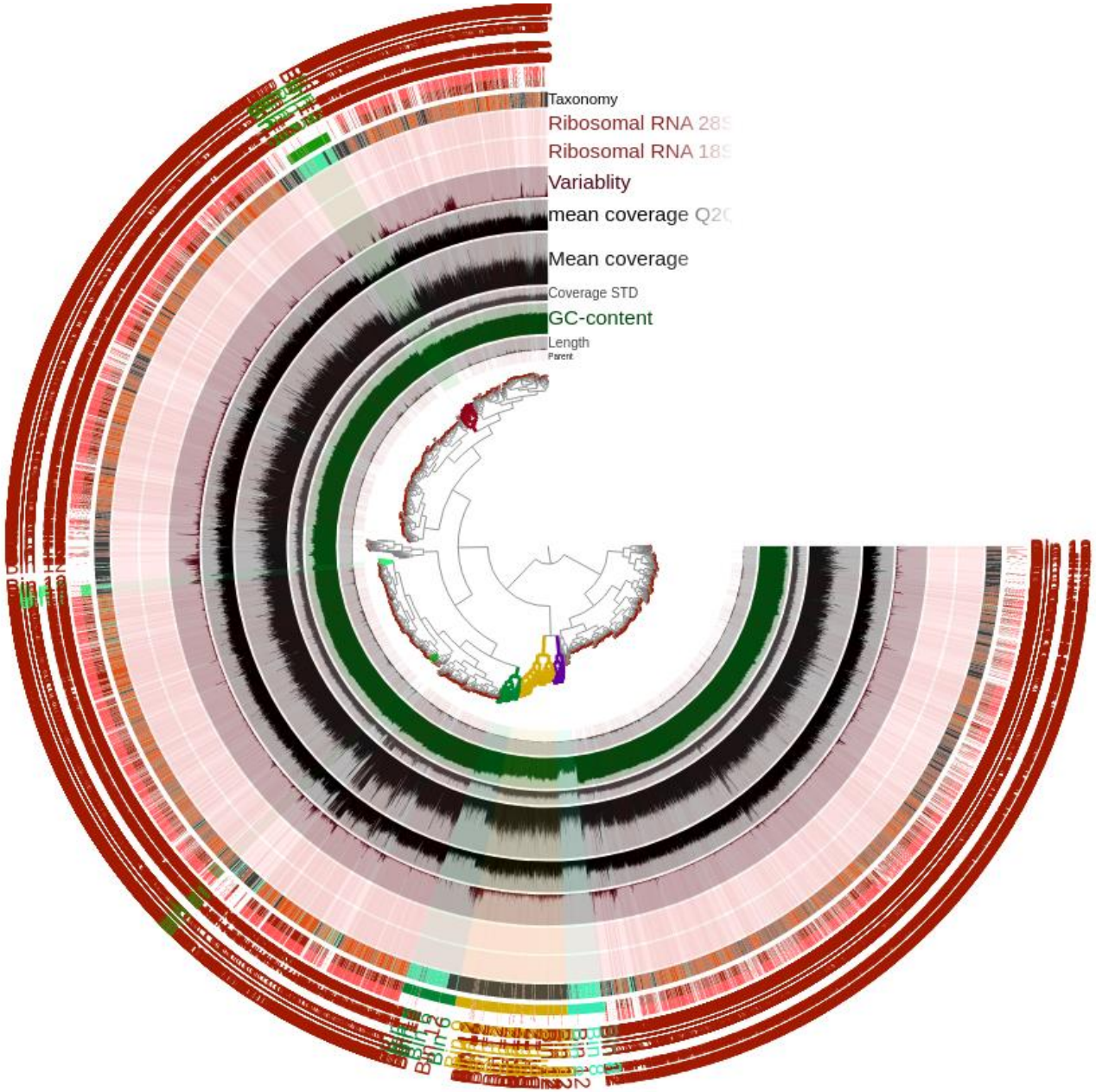


# Select bins by searching

- Field: Taxonomy
- Operator: contains / ==
- Term: Keyword



# Bin summary pages

- Contigs stats

| Contigs Stats | GK_transcriptome_200_EUnk_anvio |
|---|---|
| Total Length | 46,602,092 |
| Num Contigs | 29,137 |
| Num Contigs > 100 kb | 0 |
| Num Contigs > 50 kb | 0 |
| Num Contigs > 20 kb | 14 |
| Num Contigs > 10 kb | 236 |
| Num Contigs > 5 kb | 1,437 |

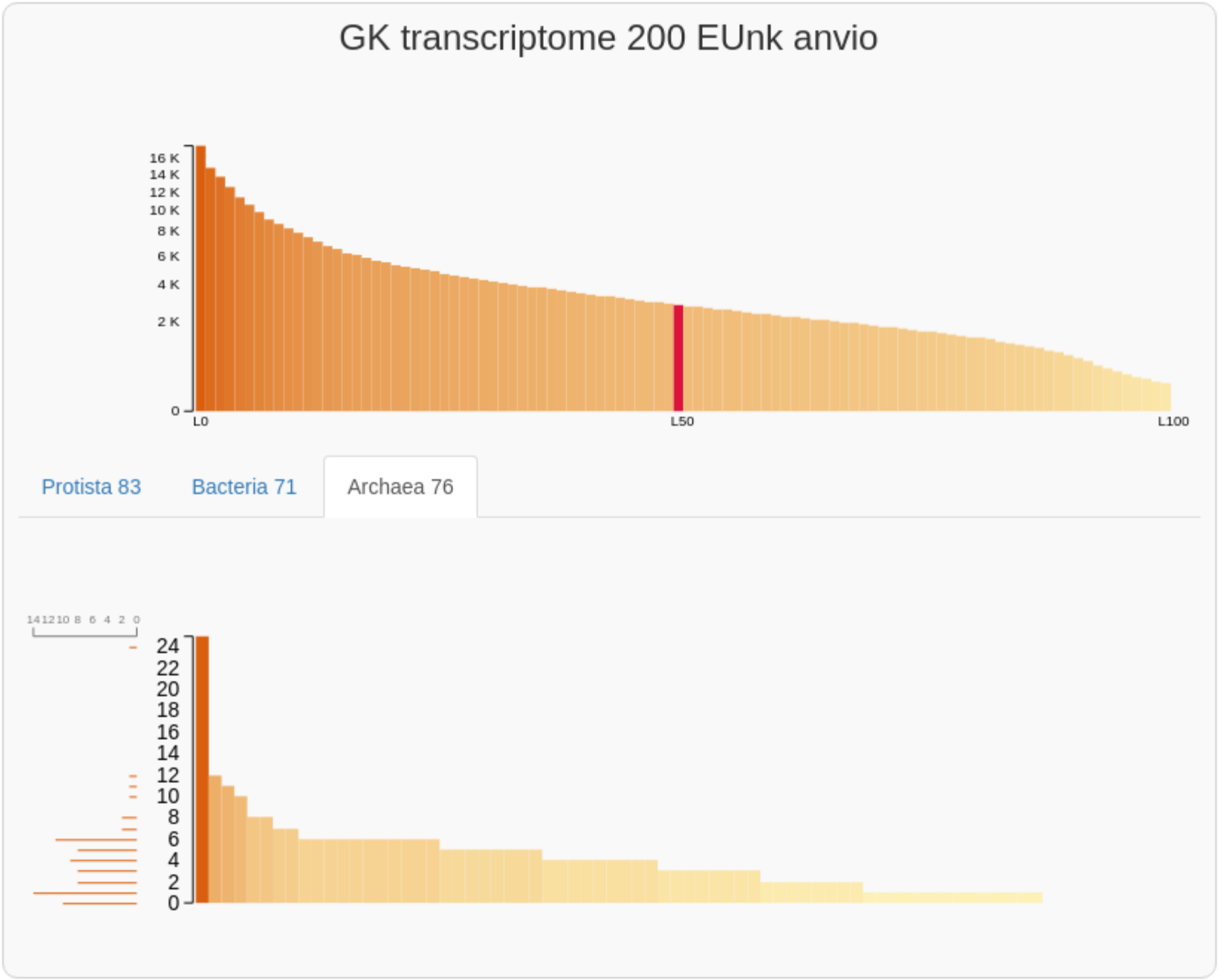
| Raw number of HMM Hits | GK_transcriptome_200_EUnk_anvio |
|---|---|
| Archaea_76 | 284 |
| Bacteria_71 | 247 |
| Protista_83 | 117 |
| Ribosomal_RNA_12S | 0 |
| Ribosomal_RNA_16S | 0 |
| Ribosomal_RNA_18S | 1 |
| Ribosomal_RNA_23S | 0 |
| Ribosomal_RNA_28S | 1 |
| Ribosomal_RNA_5S | 0 |

| Approx. number of genomes [?] | GK_transcriptome_200_EUnk_anvio |
|---|---|
| archaea (Archaea_76) | 1 |
| eukarya (Protista_83) | 1 |
| bacteria (Bacteria_71) | 1 |

| GK_transcriptome_200_EUnk_anvio |
|---|
| 5,821 |
| 35,677 |
| 201 |
| 39,614 |
| 4,804 |
| 10,286 |
| 16,239 |
| 2,810 |
| 1,601 |
| 774 |

# Bin summary pages

- L50
- can choose different hmm hits

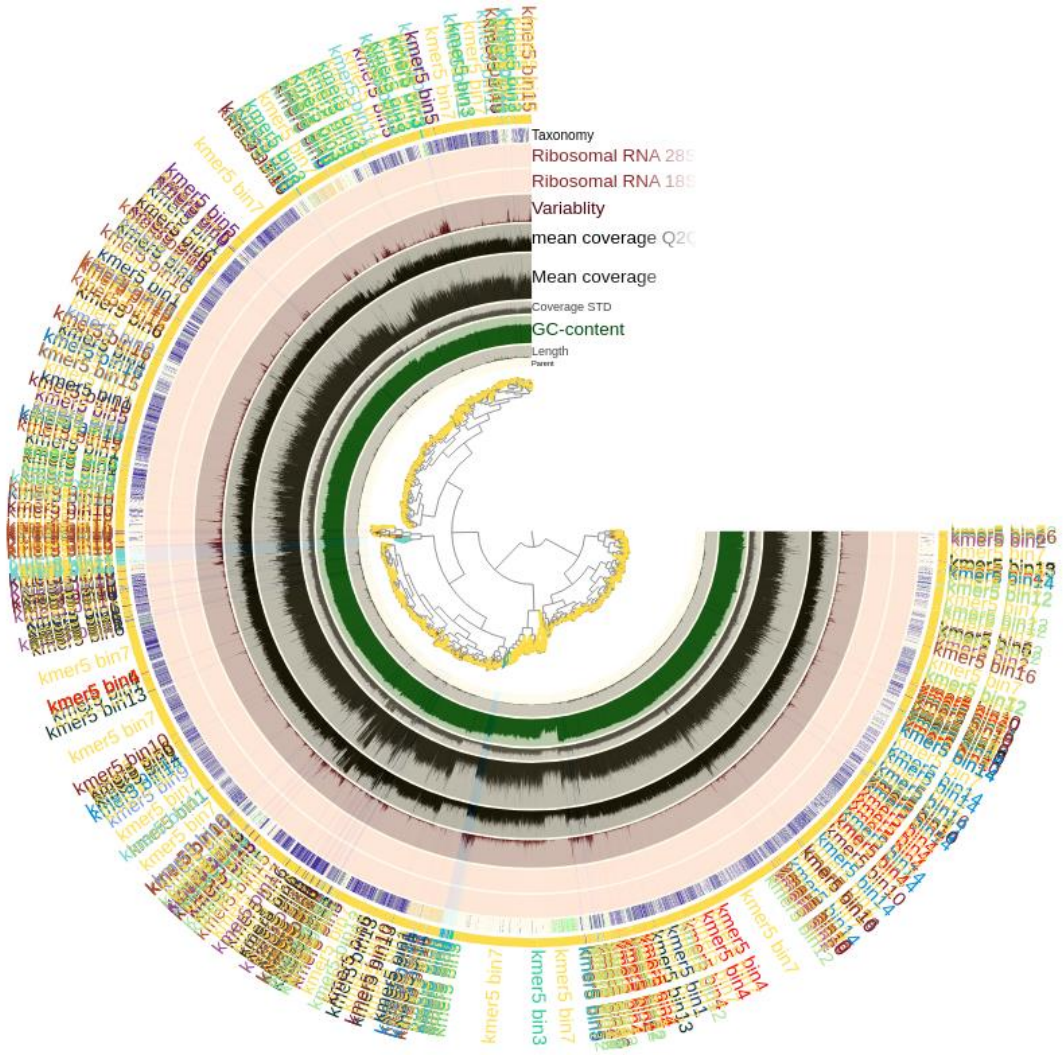


# ImportCollection (Binning results)

```
anvi-import-collection MyCC_binning_fasta.tsv \
   -p SR_ProfDir/PROFILE.db \
   -c $contigs \
   --bins-info MyCC_bin_color_table.tsv\
   -C MyCC
```

| | Name | Splits | Len | Comp. | Red. |
|---|---|---|---|---|---|
| ○ | kmer5_bin1 | 88 | 254K | ?? | ?? |
| ○ | kmer5_bin10 | 63 | 81.4K | ?? | ?? |
| ○ | kmer5_bin11 | 241 | 148K | ?? | ?? |
| ○ | kmer5_bin12 | 135 | 410K | ?? | ?? |
| ○ | kmer5_bin13 | 37 | 99.8K | ?? | ?? |
| ○ | kmer5_bin14 | 66 | 117K | ?? | ?? |
| ○ | kmer5_bin15 | 117 | 71.8K | ?? | ?? |
| ○ | kmer5_bin16 | 78 | 85.2K | ?? | ?? |
| ○ | kmer5_bin17 | 50 | 214K | ?? | ?? |
| ○ | kmer5_bin2 | 40 | 124K | ?? | ?? |
| ○ | kmer5_bin3 | 127 | 90.0K | ?? | ?? |
| ○ | kmer5_bin4 | 49 | 141K | ?? | ?? |
| ○ | kmer5_bin5 | 69 | 90.8K | ?? | ?? |
| ○ | kmer5_bin6 | 104 | 309K | ?? | ?? |
| ○ | kmer5_bin7 | 21453 | 42.6M | 96.4 | 32.5 |
| ○ | kmer5_bin8 | 144 | 89.8K | ?? | ?? |
| ◉ | kmer5_bin9 | 59 | 200K | ?? | ?? |

SETTINGS



# Important Notes:

- keep using the same version when running all the shells and when using the GUI for anvi-interactive
- Rerun Anvio, the assembly header names might change.
  - ✓Use new folder for each run
  - ✓Backup profile & database files if trying new things
- Interactive interface is super slow, click and wait, be PATIENT
- For transcriptome binning,; :
  - ✓Use Bowtie2 for mapping
  - ✓Avoid HiSat2 due to incompatibility with Anvi'o
- Functional annotation:
  - ✓Check for new version of databases: `interproscan`

# Advantages and Limitations of Using Anvi'o

**Pros:**

1. Visual Interactive Interface
    - Clear visualization of contigs based on coverage, GC content, taxonomy, and more
2. Manual Refinement
3. Flexible Inputs
4. Integration with Other Tools
5. Genome Statistics
    - Anvi'o calculates GC content, N50, coverage stats, and provides tools to export bins for downstream validation with BUSCO or QUAST
6. Customizable Workflow
    - Combine different binning strategies (e.g., supervised with taxonomy + unsupervised with tetranucleotide frequency)
    - Especially powerful for datasets with multiple eukaryotic genomes

**Cons:**

1. Not Optimized for Eukaryotes by Default
    - Originally developed for bacterial and archaeal binning
    - Some features (like functional annotation and HMMs) are biased toward prokaryotes
    - No native support for eukaryotic BUSCO sets — must be run externally
2. Resource-Intensive
    - Large eukaryotic assemblies (especially with lots of scaffolds) can slow down or crash the interface if memory is limited
    - Especially noticeable when binning complex metagenomes with >10k contigs
3. Learning Curve
    - Requires understanding of how to prepare input files (coverage, contigs, optional taxonomy)
    - Some users find the command line interface challenging at first
4. No Built-in Automatic Binning for Eukaryotes
    - Must manually inspect and select contigs — time-consuming but necessary for accuracy