

# MAGs assembly pipeline

A pipeline of the Vonaesch Lab for the curnagl cluster

Simon Yersin

With the help of the Sunagawa Lab, the Engel Lab, Julian Garneau and the  
NCCR Methods in Microbiomics

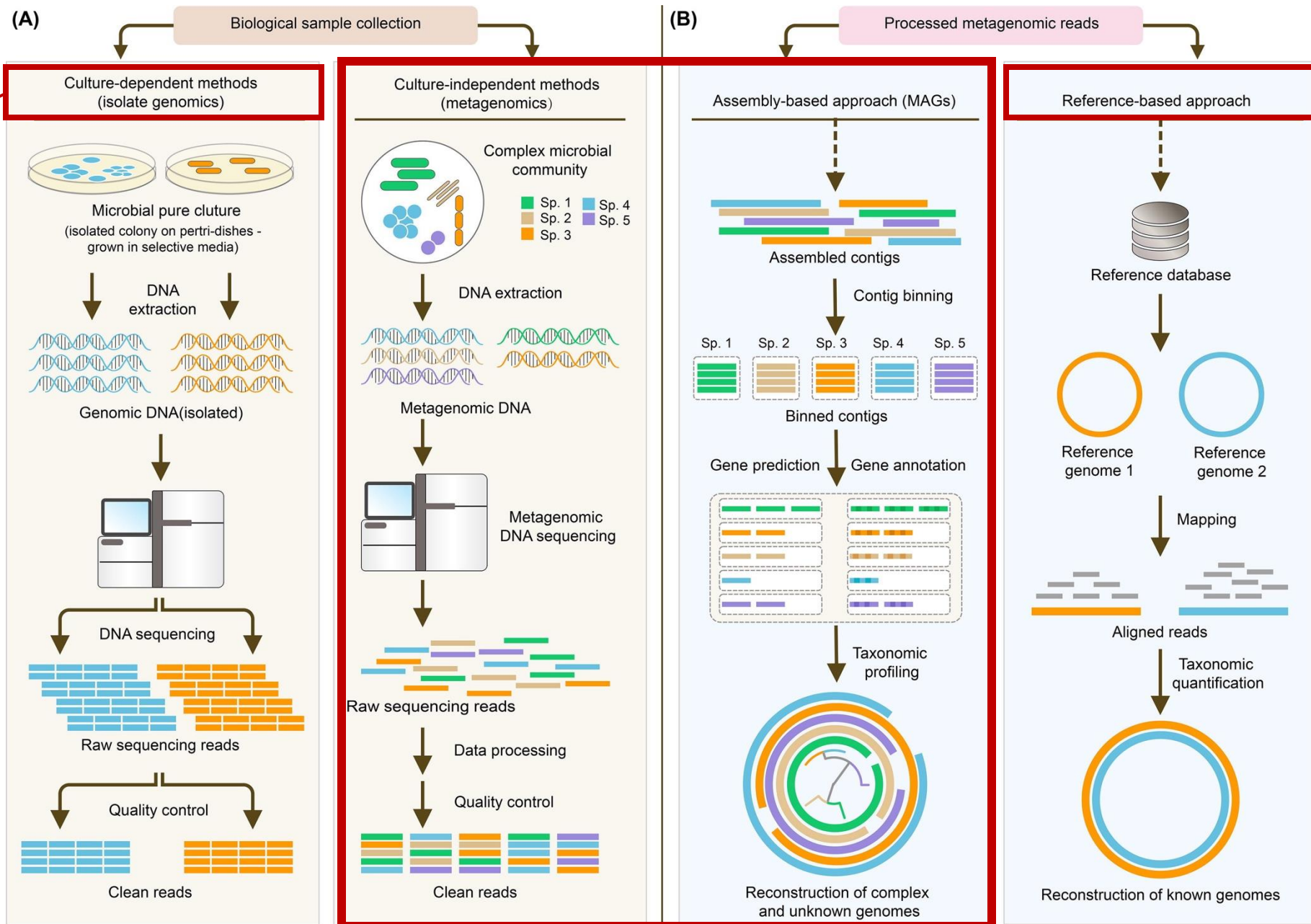
# Introduction

- This bioinformatic pipeline attempts to describe the steps to go from shotgun metagenomic sequences to metagenomic assemblies and annotated Metagenome-Assembled Genomes (MAGs).
- The pipeline include the following general steps:
  - Cleaning
  - Metagenomic assembly
  - MAGs reconstruction
  - Gene prediction and annotation

# The challenge of microbe characterization

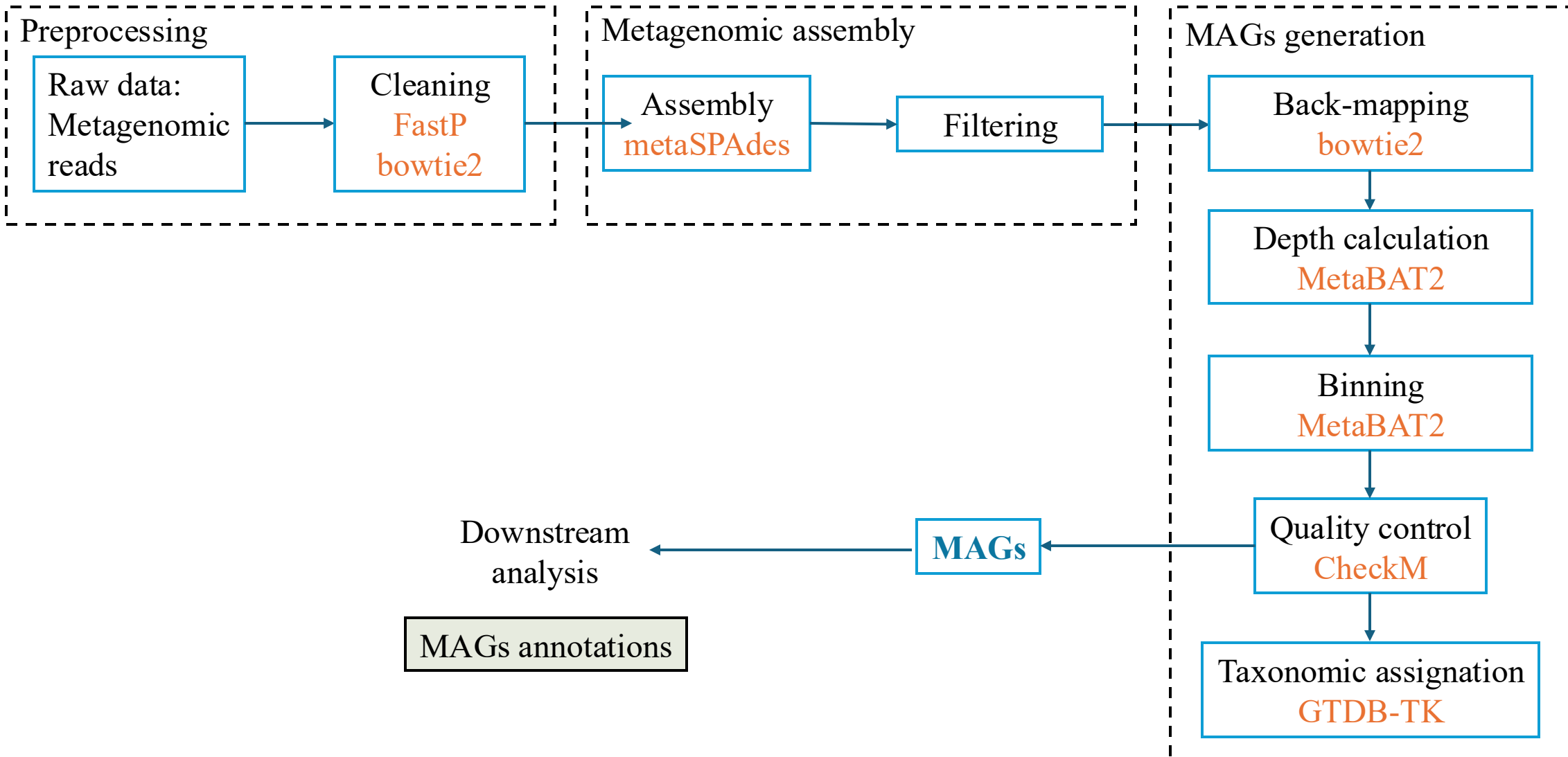
- Microbes can be characterized using traditional culture-dependent approaches that involve the isolation and sequencing of individual microbes from lab-based cultures (see isolations pipelines).
- Microbes that cannot be characterized using culture-dependent approaches, may be identified by **metagenomic sequencing**, which enables the retrieval of genomic sequences from a mixture of microbial DNA by next generation sequencing in a **culture-independent way**.
- Most previous studies on the microbiome have relied on the availability of **reference genomes**. They have involved the direct alignment of sequencing data against reference genomes, marker gene sets or species-specific sequences for taxonomic assignments. The k-mer frequencies and read depth of universal single-copy genes are commonly used to estimate taxonomic abundance. However, as **reference genomes are incomplete**, it is currently only possible to explore associations for microbes with high-quality reference genomes. Therefore, a well-characterized collection of reference genomes generated from metagenomic sequencing is required.
- This challenge has been attempted to be addressed by constructing **metagenome-assembled genomes (MAGs)**. **A MAG refers to a group of scaffolds with similar characteristics from a metagenome assembly that represent the microbial genome.** In this approach, sequencing reads are assembled into scaffolds and then the scaffolds are grouped into candidate MAGs based on tetranucleotide frequencies (TNFs), abundances, complimentary marker genes, taxonomic alignments and codon usage. The MAGs with high completeness and low levels of contamination are then used for further taxonomic annotation and gene prediction.

Isolate pipeline  
Illumina short reads  
PacBio long reads



mOTUs  
MetaPhlAn

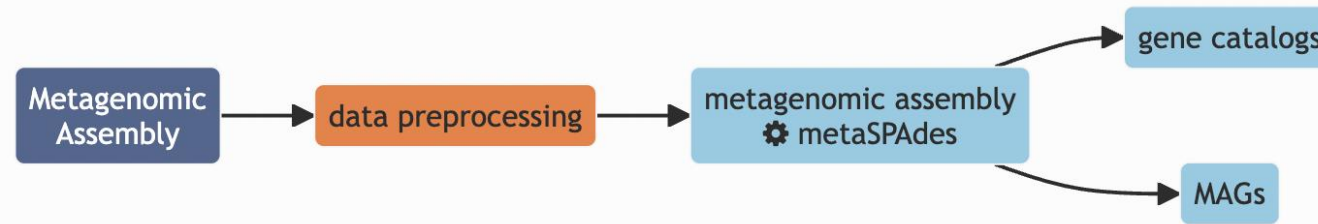
# Pipeline for MAGs construction



# MAGs assembly of paired-end reads

The blue pipeline, MAGs assembly of paired-end reads, describes the steps for the cleaning, metagenomic assembly, and MAGs construction for shotgun metagenomic sequencing data with paired forward and reverse reads.

# Metagenomic Assembly



- **Data Preprocessing.** Before proceeding to the assembly, it is important to preprocess the raw sequencing data. Data preprocessing include the standard quality control and adapter trimming, as well as host reads removal (Steps 0 and 1)
- **Metagenomic Assembly.** Following data preprocessing, we use clean reads to perform a metagenomic assembly using **metaSPAdes**. metaSPAdes is part of the SPAdes assembly toolkit. Following the assembly, we generate some assembly statistics using assembly-stats, and filter out contigs that are  $< 1$  kbp in length. The script we use for scaffold filtering can be found here: scaffold\_filter.py. It is also included in the test dataset for this section.
- The metagenomic scaffolds generated can then be used to build and/or profile Gene Catalogues or to construct MAGs.

[https://methods-in-microbiomics.readthedocs.io/en/latest/assembly/metagenomic\\_workflows.html#mags](https://methods-in-microbiomics.readthedocs.io/en/latest/assembly/metagenomic_workflows.html#mags)

## STEP 0: PREPARATION

➤ Prepare your directory with sub-directories in /user:

 Project\_name

 output

 mags\_analysis

 scripts

➤ Prepare directory and sub-directories in /scratch

 Project\_name\_scratch

 data

 output\_data

 mags\_output

 std\_output

 tmp



## STEP 0: PREPARATION

- We are assuming that the following steps have already been performed upon reception of the data:
  - Storage on the NAS and a hard drive
  - Integrity check using md5
  - Renaming of the files
- Data stored on the NAS/Recherche-P cannot be accessed from the curnagl cluster. These data can be cleaned on the supercomputer. The cleaned data, with the human reads removed, can be transferred to the NAS and copy to /scratch to be further processed.
- Copy your raw data from the NAS to scratch raw\_data folder:



```
>_ cp /nas/FAC/FBM/DMF/pvonaesc/default/D2c/YourDir/<raw data directory>  
/scratch/username/Project_name/raw_data
```
- Wait for the copy to finish and verify with **ls** that all your files are copied
- Adapt and run **create\_dir.sh** to have each sample in a separate directory

## STEP 0: PREPARATION

- **DO NOT FORGET TO ADAPT THE SCRIPTS AT EACH STEP**
- A few things to adapt:
  - #SBATCH parameters:

--job-name	--output	--cpus-per-task	--time
--error	--mem	--array	
  - Module version depending on the stack used (use module spider to check the versions)
  - Variables! Change the path
  - Control if some path are explicitly written in the script

## STEP 1: Cleaning


- These two steps are performed either on the supercomputer OR on the Urblauna cluster for sensitive data (please see Urblauna instruction)
- If no sensitive data, follow:
- Adapter and low-quality reads are cleaned with **FastP** and host reads are removed with the **bowtie2** aligner and a **reference genome** (human, mouse, or others host organisms).
- Clean your reads with **FastP** using:  
 `01_1_fastp.sh`
- Removed host reads with **bowtie2** using (only none human samples)  
 `01_2_host_reads.sh` (adapt index!)
- QC report are produced before cleaning, after FastP and after bowtie2 with **MultiQC** and can be found in data\_analysis/multiQC

## STEP 2: Assembly

- Assembly is done with **metaSPAdes** (<https://github.com/ablab/spades>). Spades produces a lot of different files, including assembly graphs, which should be kept as they can be used for other analysis.
- SPAdes is a *de novo* genome assembly (i.e. extend short reads from sequencing into contigs to reconstruct original sequence or scaffolds, *de novo* approach allows assembly without prior knowledge of the original sequences) pipeline that can deal with data from several sequencing technologies. It consists of 4 stages:
  1. Assembly graph construction
  2. k-bimer adjustment
  3. Constructs the paired assembly graph
  4. Contig construction: A contig is a DNA sequence made by assembling multiple short, overlapping sequences.
- Prior to assembly, SPAdes uses a modification of Hammer for error correction and quality trimming (computationally intensive).
- SPAdes will try constructing the **contigs** using several sizes of k-mers (21, 33, 55, 77, 99 and 127, files are temporary). The final contigs will be found in **contigs.fasta**.
- **Scaffolds** are then produced. Scaffolding is the process of ordering and orienting contigs. Scaffolds may contain gaps ('NNNN') which is not the case for contigs. Contigs placement information may come from read pairs, long read sequences and reference sequence alignment. Scaffolds are produced from the contigs and gaps using two techniques:
  - Tries to estimate the size of the gaps separating contigs using reads pairs
  - Using the assembly graph, join contigs that are separated by complex tandem repeat, that cannot be resolved exactly, with a fixed gap size of 100bp.

Scaffolding results are found in **scaffolds.fasta**.

## STEP 2: Assembly

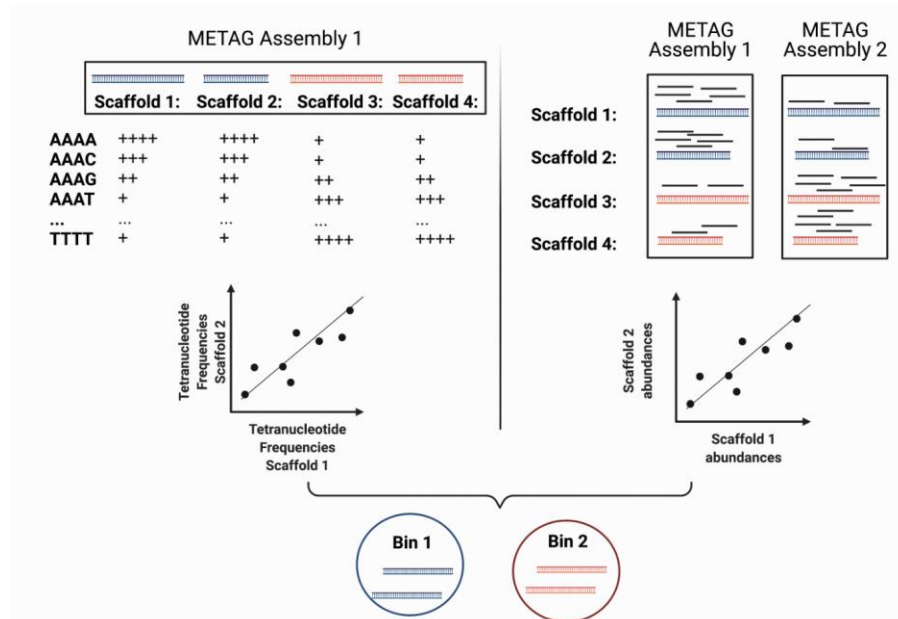
- Assemble your reads into contigs using **metaSPAdes**:  
 02\_metaspades.sh
- The following files are the output used in the next steps:
  - assembly\_graph.fastg
  - contigs.fasta
  - scaffolds.fasta
  - spades.log
- These files are copied from the /scratch directory to the /users directory in the output directory created earlier for each sample
- Additionally, several files are removed as they are not used in any of the downstream processing of the pipeline

## STEP 3: Filtering

- The filtering of the contigs and the scaffolds is done using the python script `scaffold_filter.py`.
- This step use the conda environment `biopython`. It is activated at the start of the script.
- Filter the contigs and the scaffolds using:  
`03_contig_filter.sh`
  - The script produce fasta files including the filter contig or scaffolds with different thresholds (0, 500 and 1000) and stats files. They are both copied to the `/users` directory.

# MAGs generation

- The goal of metagenomics is to be able to assemble individual microbial genomes from complex community samples. However, short-read assemblers fail to reconstruct complete genomes. For that reason, binning approaches have been developed to facilitate creation of Metagenome Assembled Genomes (MAGs).
- MAG reconstruction algorithms have to decipher which of the scaffolds generated during Metagenomic Assembly belong to the same organism (referred to as bin). While different binning approaches have been described, here we use MetaBAT2 for MAG reconstruction. As shown in the figure below, MetaBAT2 uses scaffolds' tetranucleotide frequencies and abundances to group scaffolds into bins.



- In this (very) simplified example, the blue scaffolds show similar tetranucleotide frequencies and similar abundances (across multiple samples), and consequently end up binned together, and separately from the red scaffolds.

- Metagenome-assembled genome (MAG) is a genome that is reconstructed or recovered from a metagenome (i.e. the entire DNA content of an environment). MAGs are typically reconstructed from short reads via *de novo* assembly (see assembly with SPAdes) and metagenomic binning strategies. A MAG can be a single contig or a collection of contigs (i.e. scaffolds) that, in theory, collectively represent a single microbial organism although binning can lead to errors.
- Metagenomic binning is a set of computational strategies that aims to identify and put together contigs/scaffolds that belong to the same population. These strategies often use:
  - Differential coverage of contigs (when multiple samples are present): Change in coverage (i.e. the average number of sequencing reads that map to each nucleotide position in a reference) of a reference sequence across multiple samples.
  - Sequence composition information (tetra-nucleotide frequency): The ratio of all 4-nucleotide word in a given contig. The tetra-nucleotide frequency is largely preserved throughout microbial genomes, which enables the identification of distinct contigs that likely originate from the same population.

## MAG/Bins building:

- This workflow starts with size-filtered **metaSPAdes** assembled-scaffolds (min1000). We use two software for the MAG building and binning, **bowtie2** and **MetaBAT2**. Steps:
  1. All-to-all alignment: quality-controlled reads for each of the metagenomic samples are mapped to each of the metagenomic assemblies using **bowtie2**. We select groups of max 50 samples if there are more than 50 samples. This step helps to quantify how many reads map to each contig in each sample.
  2. Within- and between-sample abundance correlation for each contig. **MetaBAT2** provides [jgi\\_summarize\\_bam\\_contig\\_depth](#) script that allows quantification of within- and between-sample abundance for each scaffold. Here we generate an abundance (depth) file for each metagenomic assembly by providing the alignment files generated using this assembly (depth is a measure of how many reads align to each contig).
  3. Metagenomic binning: running **MetaBAT2** to bin metagenomic assemblies using depth files generated in the previous step
  4. Quality control: see completeness and contamination using **CheckM**.



## STEP 4: Back mapping

- The MAGs building steps starts with size-filtered metaSPAdes assembled scaffolds ( $\geq 1000$ bp scaffolds). Reads for each of the samples are mapped to each of the assemblies using **bowtie2**.
- First, build an index for each sample.scaffold.min1000.fasta with **bowtie2** using:

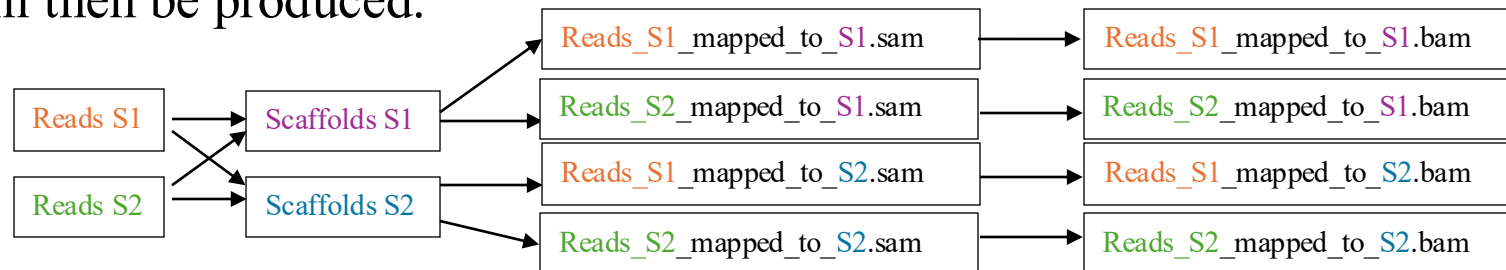
 04\_1\_index\_scaffold.sh

The indexes are stored in the **/metaspades/sample/sample\_index**

- Then, do the back mapping which consist of aligning the reads of all samples (max 50) on each sample index with **bowtie2** using:

 04\_2\_backmapping.sh

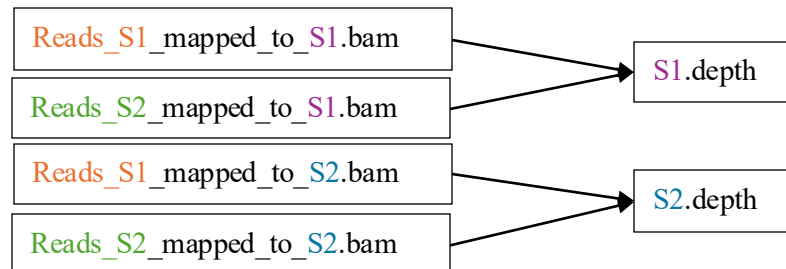
For each sample, an alignment file (.sam) is produced for each sample reads, changed to a sorted bam file and saved in **/metaspades/sample/sample\_backmapping**. If we have N sample, NxN alignment files will then be produced.



## STEP 5: Within- and between-sample abundance correlation

- **MetaBAT2** provides *jgi\_summarize\_bam\_contig\_depth* script that allows the quantification of within- and between-samples abundance for each scaffold. We generate an abundance (depth) file for each metagenomic assembly by providing the alignment file generated using this assembly. N.B. binning can be performed without this step, but it improves the MAGs quality significantly.
- Generate depth files using:

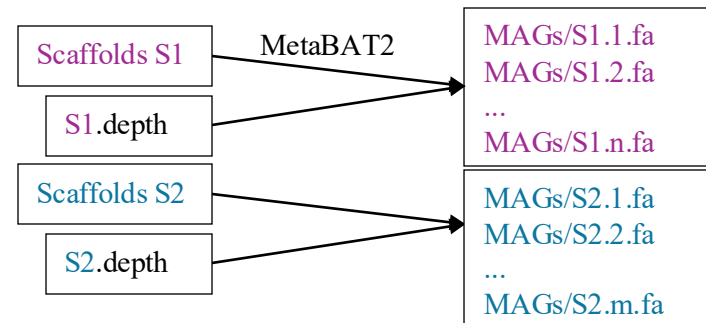
 05\_depth.sh



## STEP 6: Metagenomic Binning

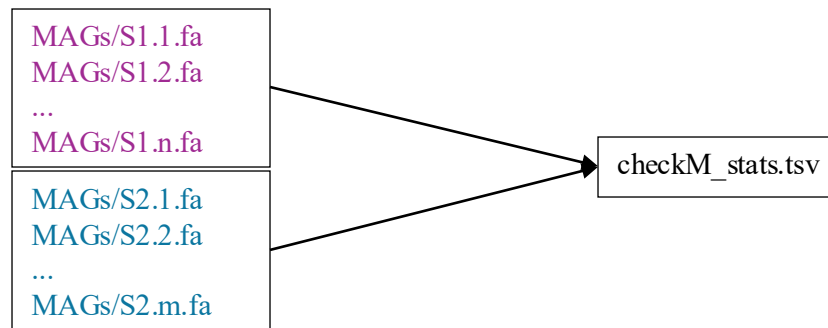
- **MetaBAT2** is the metagenome binning software tool used for efficient genome reconstruction from metagenome assemblies. **MetaBAT2** clusters metagenomic contigs into different bins, each of which should correspond to a putative genome. Input file is the contigs or scaffolds (in our case scaffolds.min1000) for each sample as fasta format and a file having mean and variance of base coverage depth. Discovered bins are saved as fasta format.
- Bin metagenomic assemblies with MetaBAT2 using the depth files generate earlier using:

 06\_binning.sh



## STEP 7: MAGs Quality Control

- Estimate how well the binning performed with CheckM using:  
📄 07\_checkM\_mags.sh
  - CheckM produce a report `checkM_stats.tsv` that can be downloaded to evaluate the quality of the MAGs



- High quality MAGs (completeness >80% and contamination <10%) can be selected for downstream analysis

## STEP 8: MAGs Taxonomic classification

- Taxonomic classification of the MAGs is determined using GTDB-TK:

 08\_gtdbtk\_mags.sh

- GTDB-Tk uses HMMER to identify marker genes in the reference genomes from the Genome Taxonomy Database (GTDB). These marker genes are concatenated for multiple-sequence alignment (MSA) and then used to construct a reference tree using a likelihood-based phylogenetic inference algorithm. GTDB-Tk performs taxonomic classification for a queried MAG based on its position in the reference tree, relative evolutionary divergence and average nucleotide identity to the reference genome.

## STEP 9: MAGs dereplication (optional)

- DerePLICATE the MAGs into Strain-level Genome Bins using dREP:

 09\_drep.sh

- dRep output include:

**data\_tables** folder with several csv containing informations about the genomes.

- **Widb.csv** file is the table with the genome score, completeness and contamination for the “winning” genomes representing the dereplicated bin.
- **Wdb.tsv** file is the table with the winning genomes, cluster and score.
- **Cdb.csv** file is the table with all the genomes and their cluster designation

The figures folder contains the dendrograms for the clustering of the genomes, the cluster scoring figures, the clustering scatterplots (alignment statistics) and information about the winning genomes.

The dereplicated\_genomes folder contains all the ”winning” genome files (.fa).

## STEP 10: Compute coverage statistics (optional)

- CoverM genome calculate the coverage of a set of reads on a set of genomes.

<https://github.com/wwood/CoverM>

- Three options are possible (more are available on the github):
  - Use **trimmed\_mean** to compute the average coverage of each position in the genome: Average number of aligned reads overlapping each position after removing the most deeply and shallow-ly covered position
  - Use **covered\_fraction** to calculate the fraction of the genome covered by mapped reads. Using min-covered-fraction 0 includes low coverage.
  - Use **count** to count the number of reads that align to each genome. It provides an absolute count of mapped reads for each genomic region. Useful for determining the abundance of specific genomes or region based on the number of reads that map to them.
- Use bwa-mem method
- discard all the MAGs with <50% coverage breadth
- Option: use dereplicated MAGs at the species level
- CoverM result is a .tsv file for each sample.

## MAGs analysis and selection

- Before the downstream analysis, we can select our MAGs of interest. Import the checkM and gtdb table into RStudio, merge the table and filter to select the MAGs of interest based on the following criteria:
  - Completeness  $\geq 80\%$  OR  $\geq 50\%$
  - Contamination  $\leq 5\%$  OR  $\leq 10\%$
  - Taxonomy
- Create a config.txt file with the first column being ArrayTaskID and second the name of the MAGs of interest
- Use [copy\\_HQ\\_mags.sh](#) to copy the MAGs of interest into a new folder. Then, move to the downstream analysis: MAGs annotations



# MAGs assembly of single-end reads

The orange pipeline, MAGs assembly of single-end reads describes the steps for the cleaning, metagenomic assembly, and MAGs construction for shotgun metagenomic sequencing data with single forward reads.


Currently, metaSPAdes do not support single-end reads for metagenomic assembly. For single-end reads, we use MEGAHIT which support both paired-end and single-end reads.

## STEP 0: PREPARATION

➤ Prepare your directory with sub-directories in /user:

 Project\_name

 output

 mags\_analysis

 scripts


➤ Prepare directory and sub-directories in /scratch

 Project\_name

 data



 output\_data

 mags\_output

 std\_output

 tmp

## STEP 1: Cleaning

- Adapter and low-quality reads are cleaned with **FastP** and host reads are removed with the **bowtie2** aligner and a **reference genome** (human, mouse, or others host organisms).
- Clean your reads with **FastP** using:  
 `01_1_fastp.sh`
- Removed host reads with **bowtie2** using:  
 `01_2_host_reads.sh`
- QC report are produced before cleaning, after FastP and after bowtie2 with **MultiQC** and can be found in `data_analysis/multiQC`

## STEP 2: Metagenomic assembly

- SPAdes can only assemble paired-end reads, for single-end reads, we use **MEGAHIT**
- Assemble your single-end reads into contigs using **MEGAHIT**:  
[02\\_megahit\\_single.sh](#)
- The following files are the output used in the next steps:  
**Sample.contigs.fa**  
**sample.log**
  - These files are copied from the /scratch directory to the /users directory in the output directory created earlier for each sample
  - Additionally, the intermediary contigs are removed as they are not used in any of the downstream processing of the pipeline
  - N.B. MEGAHIT do not produce scaffolds.

## STEP 3: Filtering

- The filtering of the contigs is done using the python script `scaffold_filter.py`.
- Filter the contigs using:
  - `03_contig_filter.sh`
  - The script produce fasta files including the filter contig with different thresholds (0, 500 and 1000) and stats files. They are both copied to the /users directory.

## STEP 4: Back mapping

- The MAGs building steps starts with size-filtered MEGAHIT assembled contigs ( $\geq 1000$ bp contigs). Reads for each of the samples are mapped to each of the assemblies using **bowtie2**.
- First, build an index for each sample.contigs.min1000.fasta with **bowtie2** using:

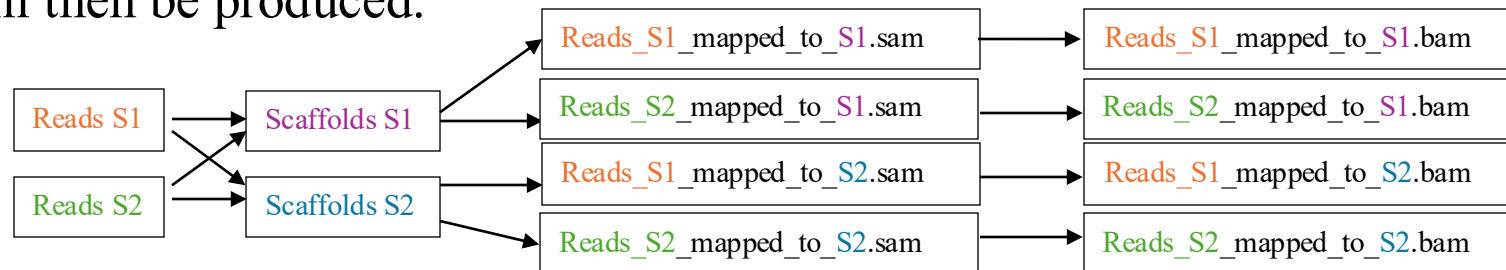
 04\_1\_index\_scaffold.sh

The indexes are stored in the **/megahit/sample/sample\_index**

- Then, do the back mapping which consist of aligning the reads of all samples (max 50) on each sample index with **bowtie2** using:

 04\_2\_backmapping.sh

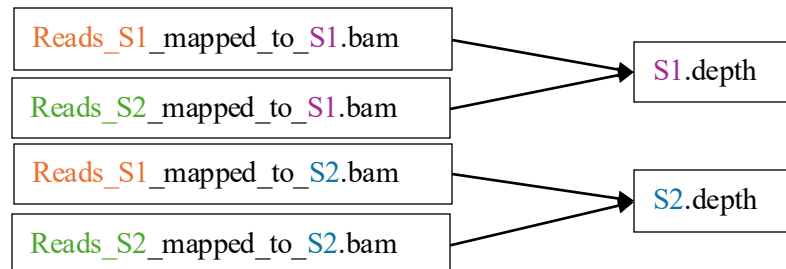
For each sample, an alignment file (.sam) is produced for each sample reads, changed to a sorted bam file and saved in **/megahit/sample/sample\_backmapping**. If we have N sample, NxN alignment files will then be produced.



## STEP 5: Within- and between-sample abundance correlation

- **MetaBAT2** provides *jgi\_summarize\_bam\_contig\_depth* script that allows the quantification of within- and between-samples abundance for each scaffold. We generate an abundance (depth) file for each metagenomic assembly by providing the alignment file generated using this assembly. N.B. binning can be performed without this step, but it improves the MAGs quality significantly.
- Generate depth files using:

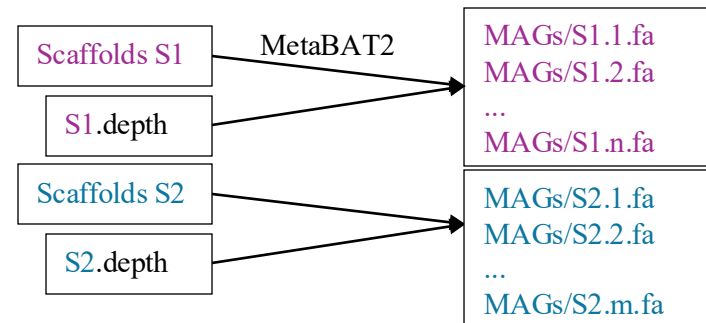
 05\_depth.sh



## STEP 6: Metagenomic Binning

- **MetaBAT2** is the metagenome binning software tool used for efficient genome reconstruction from metagenome assemblies. **MetaBAT2** clusters metagenomic contigs into different bins, each of which should correspond to a putative genome. Input file is the contigs (in our case contigs.min1000) for each sample as fasta format and a file having mean and variance of base coverage depth. Discovered bins are saved as fasta format.
- Bin metagenomic assemblies with MetaBAT2 using the depth files generate earlier using:

 06\_binning.sh



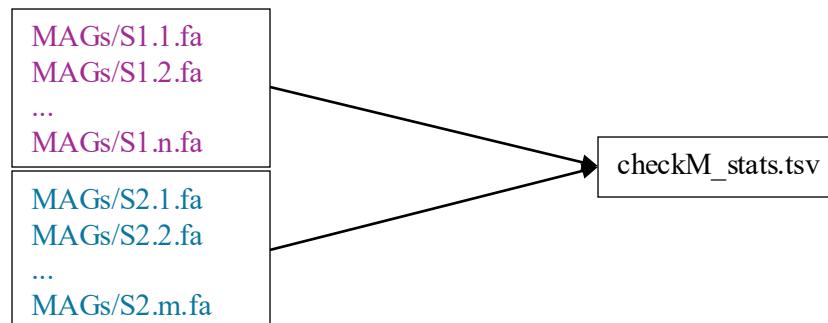


## STEP 7: MAGs Quality Control

- Estimate how well the binning performed with CheckM using:

 07\_checkM\_mags.sh

- CheckM produce a report **checkM\_stats.tsv** that can be downloaded to evaluate the quality of the MAGs



- High quality MAGs (completeness >80% and contamination <10%) can be selected for downstream analysis

## STEP 8: MAGs Taxonomic classification

- Taxonomic classification of the MAGs is determined using GTDB-TK:

 `gtdbtk_mags.sh`

- GTDB-Tk uses HMMER to identify marker genes in the reference genomes from the Genome Taxonomy Database (GTDB). These marker genes are concatenated for multiple-sequence alignment (MSA) and then used to construct a reference tree using a likelihood-based phylogenetic inference algorithm. GTDB-Tk performs taxonomic classification for a queried MAG based on its position in the reference tree, relative evolutionary divergence and average nucleotide identity to the reference genome.

## MAGs analysis and selection

- Before the downstream analysis, we need to select our MAGs of interest. Import the checkm and gtdb table into RStudio, merge the table and filter to select the MAGs of interest based on the following criteria:
  - Completeness  $\geq 80\%$
  - Contamination  $\leq 10\%$
  - Taxonomy
- Create a config.txt file with the first column being ArrayTaskID and second the name of the MAGs of interest
- Use [09\\_copy\\_HQ\\_mags.sh](#) to copy the MAGs of interest into a new folder. Then, move to the downstream analysis: MAGs annotations