

Profiling Strain-Level Diversity pipeline

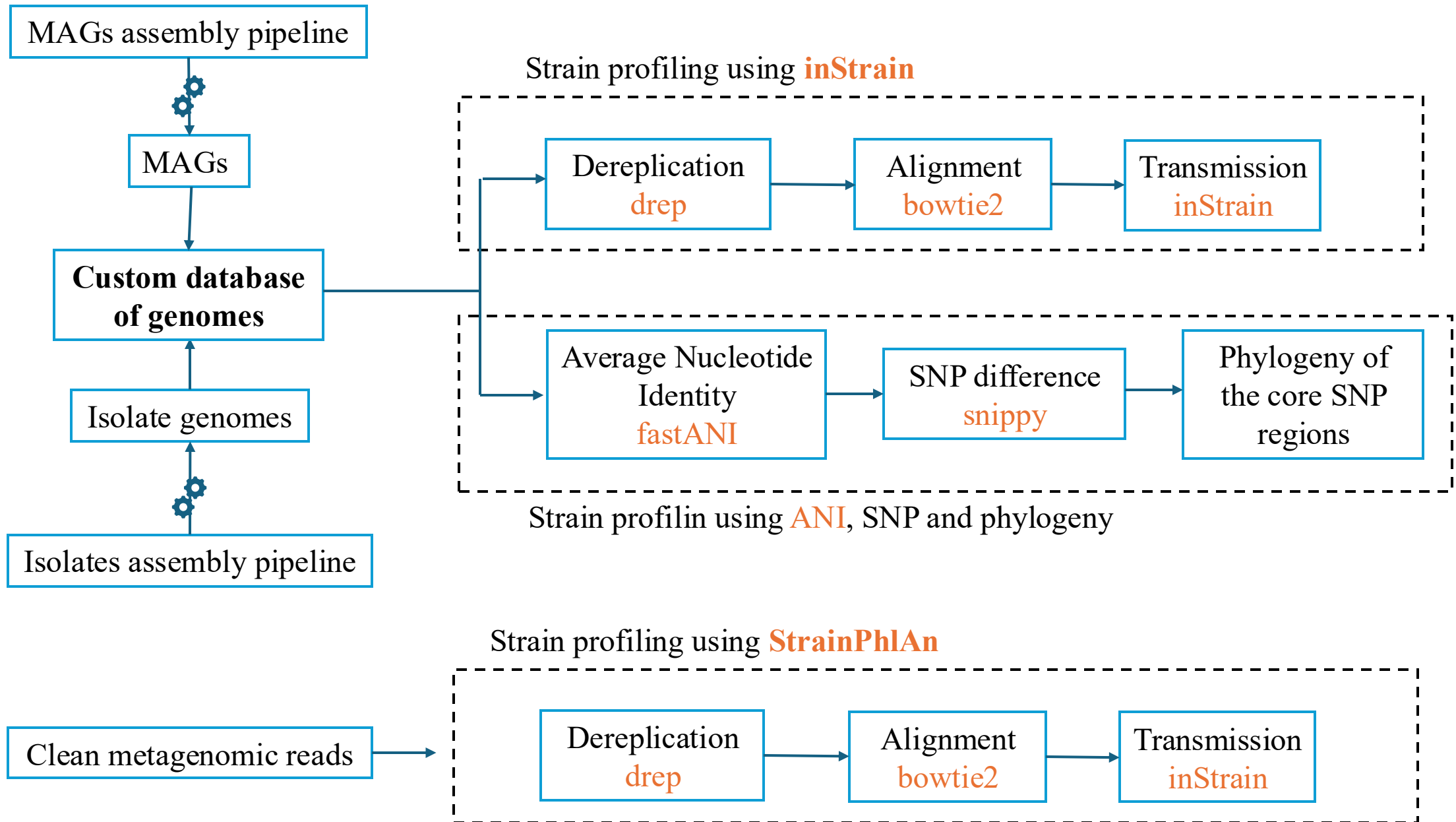
Bacterial strain transmission

A pipeline of the Vonaesch Lab for the curnagl cluster

Simon Yersin & Margaux Creze

Profiling strain-level diversity

- The Average Nucleotide Identity (ANI) is a measure of nucleotide-level genomic similarity between the coding regions of two genomes.
- Bacterial species: bacteria with an Average Nucleotide Distance (ANI) >95% tends to recombine more often with each other, which make the >95% ANI the gold standard to define a species.
- For strains, the definition is less clear (<https://www.nature.com/articles/s41579-020-0368-1>) and depends on the research question. An ANI distance of 99.9% usually indicate strains that are highly similar.
- Several types of analysis and approaches can be taken to assess for the strain-level diversity (K-mer, gene based, ...). We are interested here in the genome-based analysis which align reads to genome and calculate genome and gene metrics. Using shotgun metagenomic sequencing data, the analysis starts with MAGs reconstruction and, if possible, with isolate genomes assembled from whole genome sequencing data.



Evaluate transmission of strains using the
whole genome sequencing data

UNCOMPLETE!

Evaluate transmission of strains using the whole genome sequencing data

- Calculate the pairwise ANI distance between genomes using **fastANI**.
- Align the paired trimmed reads of the sample of interest to a species-specific reference strain using snippy.
 - snippy-core
 - snippy-clean_full_aln
- Use Gubbins to identify and remove putative areas of recombination with: -m 4 -b 4000 -first-tree-builder fasttree
- Identify SNP sites with snp-site
- Phylogeny of SNP core region → resulting file from snp-sites
- Generate phylogenetic tree with RAxML (20 maximum likelihood trees and 100 bootstrap trees)

Strain-level profiling with inStrain

Strain-level profiling using inStrain

- This pipeline starts with MAGs reconstructed using the MAGs assembly pipeline.
- Starting with a database of reconstructed MAGs, the first step is to filter for high quality MAGs by using checkM results for completeness ($>75\%$) and contamination ($<10\%$) (these thresholds can be adjusted).
- The high-quality MAGs are stored together in a single folder for the following dereplication step.
- CheckM's completeness and contamination can also be provided in the next step, but the table needs to be in the following format:

.csv file

3 columns: genome, completeness, contamination

column genome: name of the MAGs file to derePLICATE

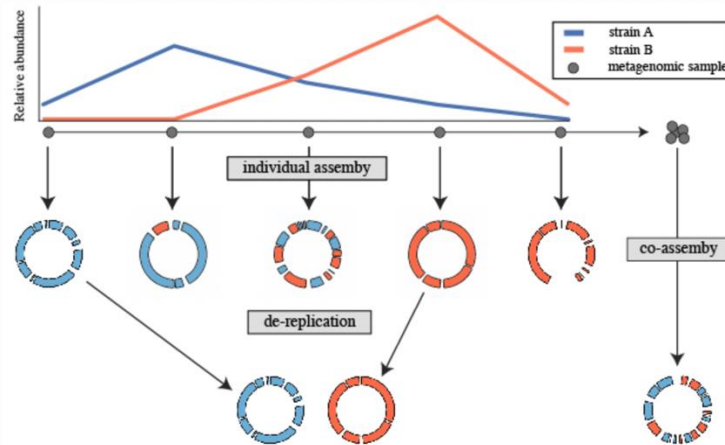
```
genome,completeness,contamination
AGN001FE.1.fa,27.99,0
AGN001FE.10.fa,73.41,1.02
AGN001FE.100.fa,78.79,3.45
AGN001FE.101.fa,29.66,0
AGN001FE.102.fa,27.19,0
AGN001FE.103.fa,11.41,0
AGN001FE.104.fa,42.65,0.77
```

Step 0: Preparation

- In the scratch, prepare the following directories architecture:
 - Project_directory
 - std_output
 - transmission_dir
 - dereplication
 - inStrain
 - instrain_profile
 - instrain_compare
 - MAGs
 - MAGs_per_samples
 - mapping_output
 - reads
 - tables
 - genome_db
 - prodigal

Step 1: Dereplication

- Dereplication is the process of identifying sets of genomes that are the “same” in a list of genomes, removing all redundant genomes that are highly similar and keeping the “best” genome for each redundant set. It results to a unified reference database representing strain diversity.
- <https://drep.readthedocs.io/en/latest/overview.html>
- By using an ANI threshold of $>95\%$, MAGs will be dereplicated at the species level. The collection of MAGs with $>95\%$ ANI is then referred as a Species-Level Genome Bin (SGB). Using $>85\%$ ANI, a set of MAGs will be referred as a Genus-Level Genome Bin and $>70\%$ ANI, a Family-Level Genome Bin.
- For strain-level diversity profiling, the ANI threshold is set at $>98\%$



Step 1: Dereplication

- Dereplicate the MAGs into Strain-level Genome Bins using dRep:

 `01_1_drep_strains.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). These genome files will be used for the next steps.

Step 2: inStrain preparation

- A few steps are necessary before being able to run inStrain.
1. InStrain can be used with different reference genome database (public or own generated database), combine the dereplicated genomes from dRep and other reference genomes into a single fasta file:

```
 cat ../dereplicated_genomes/*.fa > ../genomes_db/allGenomes_v1.fasta
```

2. Create a scaffold origin files using **dRep script parse_stb.py** (activate dRep conda environment)

```
 parse_stb.py --reverse -f ../dereplicated_genomes/*.fa -o genomes.stb
```

3. Build a bowtie2 index of the allGenomes fasta file using:

```
 02_1_building_index.sh
```

4. Map sample reads to the bowtie2 index using:

```
 02_2_map_ref_gen.sh
```

5. Run prodigal for gene-level profiling (optional) using:

```
 02_3_prodigal.sh
```

InStrain – Overview

InStrain (<https://instrain.readthedocs.io/en/latest/index.html>) is a program for microbial metagenomic analysis. When a microbial genome is sequenced, a population of cells is sequenced and there is always real biological genetic heterogeneity within a population. Every cell does not have the same genotype at every single position. InStrain can determine organism presence/absence in a community, measure and interrogate the genetic heterogeneity in microbial population, and perform detailed comparisons between organisms in different samples.

A community is a collection of taxa in a metagenome. After mapping your metagenomic reads to a set of representative genomes, inStrain can generate a number of metrics that help understand **community composition**. These include the percentage of reads that map to your representative genome database, the abundance of each microbe in the community, and a detailed picture of the organisms that are present or absent.

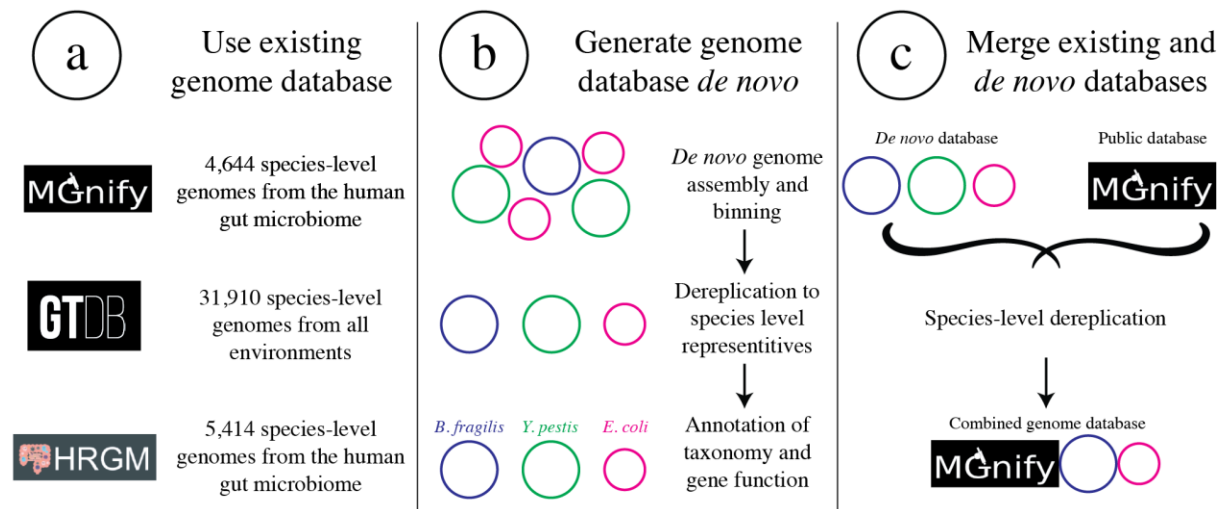
A population is the collection of cells that make up an individual taxa in a community. After mapping your metagenomic reads to a set of representative genomes, inStrain can generate a number of metrics characterizing the **population-level diversity** of each detected organism. These metrics include nucleotide diversity, SNSs and SNVs, linkage, pN/pS, iRep, and others. Most metrics are calculated on the gene level, scaffold level, and genome level.

Strain-level comparisons between populations in different communities are notoriously difficult to perform with high accuracy. After profiling the communities of metagenomic samples, inStrain can **compare the populations in the different communities** in a highly-accurate manner by taking into account the population-level diversity. This analysis reports comparison metrics including the percentage of the genome covered in each sample, popANI, conNI, and the locations of all differences between strains.

InStrain – Representative genomes

Representative genomes are genomes that are chosen to represent some group of taxa and they are the base unit of inStrain-based metagenomic analysis. If one wanted to study the species-level composition of a community with inStrain, a set of representative genomes at the desired taxonomic level, is necessary.

A collection of representative genomes is referred to as a **Genome database**. Genome databases can be downloaded from public repositories (1), generated via *de novo* sequence assembly and binning (2), or a combination of the two (3). It is important to ensure that each genome in the Genome database is distinct enough from other genomes in the database to avoid mapping confusion, and by mapping to all genomes in a Genome database simultaneously one can significantly reduce the number mis-mapped reads overall.



InStrain – Representative genomes

Representative genomes are typically chosen by first clustering a set of genomes using some ANI thresholds and second picking a single genome to represent each cluster/ A good representative genome is a high quality, contiguous, shares a high degree of gene content with the taxa it is meant to represent and has a similar ANI to all genome it's meant to represent. The program **dRep** is used to pick representative genomes, and it uses a scoring system to score each genome and pick the genome with the highest score (See previous slides).

Running **inStrain profile** will generate information about each representative genome detected in the sample. This information can be used to determine how good of a fit each representative genome is to the true population that is recruiting reads from. Helpful metrics are mean read ANI, reference conANI, reference popANI, and breadth vs. expected breadth. If there are regions of the genome with much higher coverage than the rest, it is likely that that region is recruiting reads from another population (mismapped reads).

InStrain – Strain-level comparison and popANI

InStrain is able to perform detailed, accurate, microdiversity-aware **strain-level comparisons between organisms detected in multiple metagenomic samples**. This is done using the command `inStrain compare` on multiple samples that have been profiled using the command **inStrain profile**.

To understand why “microdiversity-aware” genomic comparisons are important, consider the fact that all natural microbial populations have some level of genomic heterogeneity present within them. How does one accurately compare populations that have intraspecific genetic variation? InStrain performs microdiversity-aware comparisons using the metrics popANI and conANI.

- conANI – consensus ANI: ANI calculated based on consensus sequences. Each position on the genome is represented by the most common allele and minor allele are ignored
- popANI – population ANI: performed by inStrain, it considers both major and minor alleles. If two populations share any allele at a loci, including minor alleles, it does not count as a difference when calculating popANI

Organisms in different samples that are linked by a recent transmission event should have $\geq 99.999\%$ popANI and $\geq 50\%$ percent_genome_compared (fraction of the genome that was at sufficient coverage in both samples being compared)

Step 3: inStrain profile

- Run **inStrain profile** to generate information about each representative genome detected in the samples using:

 `03_inStrain_profile.sh`

- inStrain profile needs the following arguments:
 - .sam or .bam mapping files (read mapping files, required)
 - .fasta file the the mapping file is mapped to (reference genome, required)
 - -o output prefix (default inStrain, optional)
 - -g gene file (.fna) produced with prodigal (optional)
 - -s scaffold to bin file (.stb) produced with dRep (optional)
 - --database_mode: reduce computing time for large set of genomes
 - --skip_plto_generation: reduce computing time as plots are unnecessary for compare
- The resulting directories from inStrain profile are then used as input for inStrain compare

Step 4: inStrain compare

- Run **inStrain compare** to compare genomes that have been profiled by multiple different metagenomic mappings

 `04_inStrain_compare.sh`

- inStrain compare needs the following arguments:
 - -i list of profile directories to compare. The input for the inStrain compare command consists of the directories containing the inStrain profile outputs from multiple samples. Each of these directories should have been generated by the inStrain profile command and contain various output files such as coverage, diversity, and SNV statistics.
 - -o output directory where the comparison object will be stored
 - -p number of threads to use
 - -s or --stb scaffold to file (.stb) produced with dRep
 - -ani 0.99999 genome identity set at 99.999
 - -cov 0.5 genome coverage threshold
- Output of interest: `instrain_compare_genomeWide_compare.tsv`

Strain-level profiling with StrainPhAn

Profiling strain-level diversity

- StrainPhlAn is a computational tool for tracking individual strains across a large set of samples. The input of StrainPhlAn is a set of metagenomic samples and for a given species of interest, the output is a multiple sequence alignment (MSA) file of the strains of the species of interest reconstructed from the samples. From this MSA, StrainPhlAn calls PhyloPhlAn to build the phylogenetic tree showing the phylogenetic relationship between strains identified in each sample.
- For each sample, StrainPhlAn is able to identify strains of a species by merging and concatenating all reads mapped against species-specific MetaPhlAn markers.
- StrainPhlAn can distinguish between different strains of the same microbial species, providing a finer granularity in microbial community analysis
- It reconstructs phylogenetic trees of microbial strains based on their genomic markers, allowing researchers to study evolutionary relationships and strain-level variations
- Utilizes consensus markers derived from MetaPhlAn's profiling to perform the strain-level analysis
- Example of cohorts that used StrainPhlAn for strain transmission
 - MicrobeMom cohort (Feehily et al, 2023)

How StrainPhlan works

1. Input Preparation:

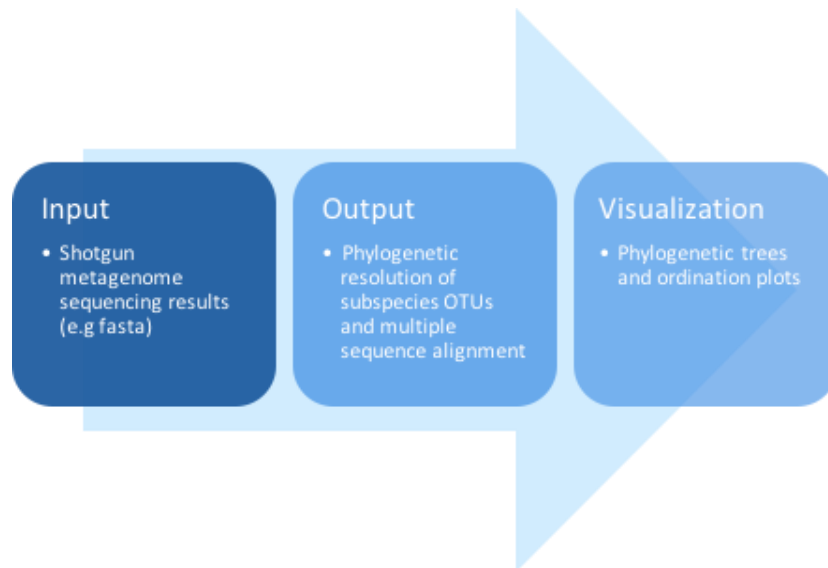
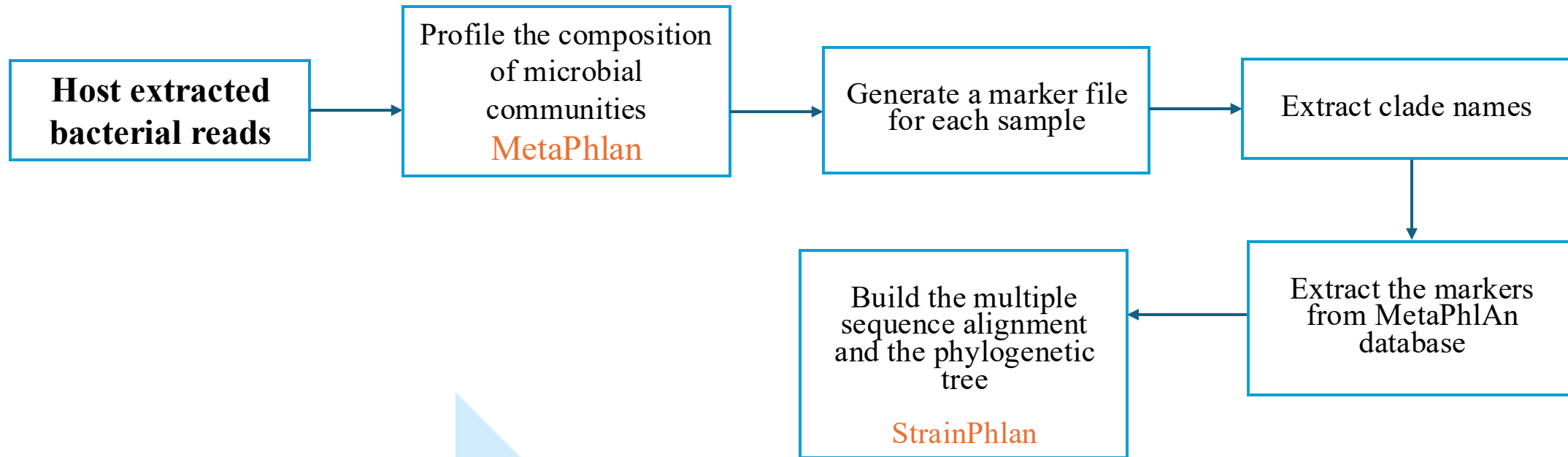
1. Consensus Markers: StrainPhlAn uses consensus markers generated from metagenomic samples. These markers are representative sequences of specific microbial strains found in the samples.
2. Clade-specific Markers: For detailed analysis, clade-specific marker genes are used to focus on particular microbial clades.
3. Reference Genomes: Reference genomes provide a basis for comparison and help in phylogenetic reconstruction.

2. Phylogenetic Analysis:

1. StrainPhlAn aligns the consensus markers against the reference genomes.
2. Constructs a phylogenetic tree to visualize and analyze the relationships between different strains.

3. Output:

1. The output includes phylogenetic trees, strain-specific profiles, and other data that can be used to interpret the strain-level diversity in the microbial community.



Step 1: MetaPhlan

Taxonomic profiling with MetaPhlan

Aim: Obtain the sam files from your samples by mapping them against MetaPhlan database

This step will run MetaPhlan: the metagenomic samples will be mapped against the MetaPhlan marker database and then SAM files (*.sam.bz2) will be produced. Each SAM file (in SAM format) contains the reads mapped against the marker database of MetaPhlan.

 [01_sam_metaphlan.sh](#)

After this step, you will have a folder "sams" containing the SAM files (*.sam.bz2) and other MetaPhlan output files in the "bowtie2" and "profiles" folders.

Step 2: Extracting the sample markers

Extracting the samples markers

Aim: Produce the consensus-marker files which are the input for StrainPhlAn

For each sample, this step will reconstruct all species strains found in it and store them in a json file (*.json). Those strains are referred as *sample-reconstructed strains*.

 [02_cons_markers.sh](#)

The result is the same if you want to run several sample2markers.py scripts in parallel with each run for a sample (this may be useful for some cluster-system settings). After this step, you will have a folder consensus_markers containing all sample-marker files (*.json).

Extract clade names

From the terminal, in the consensus_marker folder.

First activate strainphlan environment:

```
conda activate /work/FAC/FBM/DMF/pvonaesc/vonasch_lab_general/syersin/MetaPhlan/metaphlan
```

Then, run the following to extract the clade names and store into a file:

```
strainphlan -s ./*.json.bz2 --print_clades_only -o . --nproc 8 > ./clades.txt
```

Count the number of clades by running:

```
awk -F'\t' 'NR>1 {print $1}' print_clades_only.tsv | wc
```

The returned number will be the number of array for the next script


Step 3: Multiple Sequence Alignment

Aim 1: Extract the markers of some strains of interest (or all) from MetaPhlAn database (to add its reference genome later)

Produce .fna file for each marker.

Aim 2: Build the multiple sequence alignment and the phylogenetic tree

This step will filter the selected clade markers based on their presence in the *sample-reconstructed strains* and *reference-genomes* (if specified). Also, the *sample-reconstructed strains* and *reference-genomes* will be filtered based on the presence of the selected clade markers. From this filtered markers and samples, StrainPhlAn will call PhyloPhlAn to produce a multiple sequence alignment (MSA) to then build the phylogenetic tree.

 [03_msa.sh](#)

After this step, you will find the different trees in output folder. You'll have one tree per SGB marker.
Results of interest: bestTree files

StrainPhlAn results analysis

Analyse results in RStudio, we decided to use the method presented in Feehily et al, 2022 which normalise the best tree for each clade, compute the distance between sample using the ape package and then using a 0.001 threshold, decide on a sharing event between samples.

Optional: Add Metadata

In order to add the metadata, they provide a script called [add_metadata_tree.py](#)

```
add_metadata_tree.py -t output/RAxML_bestTree.t__SGB1877.StrainPhlAn4.tre -f fastq/metadata.txt -m  
subjectID --string_to_remove .fastq.bz2
```

The script "add_metadata_tree.py" can accept multiple metadata files (space-separated, wild card can also be used) and multiple trees. A metadata file is a tab-separated file where the first row is the meta-headers, and the following rows contain the metadata for each sample. Multiple metadata files are used in the case where your samples come from more than one dataset and you do not want to merge the metadata files.

For more details of using "add_metadata_tree.py", please see its help (with option "-h")

Note that "sampleID" is a compulsory field.

After adding the metadata, you will obtain the tree files "*.tre.metadata" with metadata and view them by [Archaeopteryx](#) as in the previous step.

If you have installed [graphlan](#), you can plot the tree with the [plot_tree_graphlan.py](#) script:

```
plot_tree_graphlan.py -t output/RAxML_bestTree.t__SGB1877.StrainPhlAn4.tre.metadata -m subjectID
```