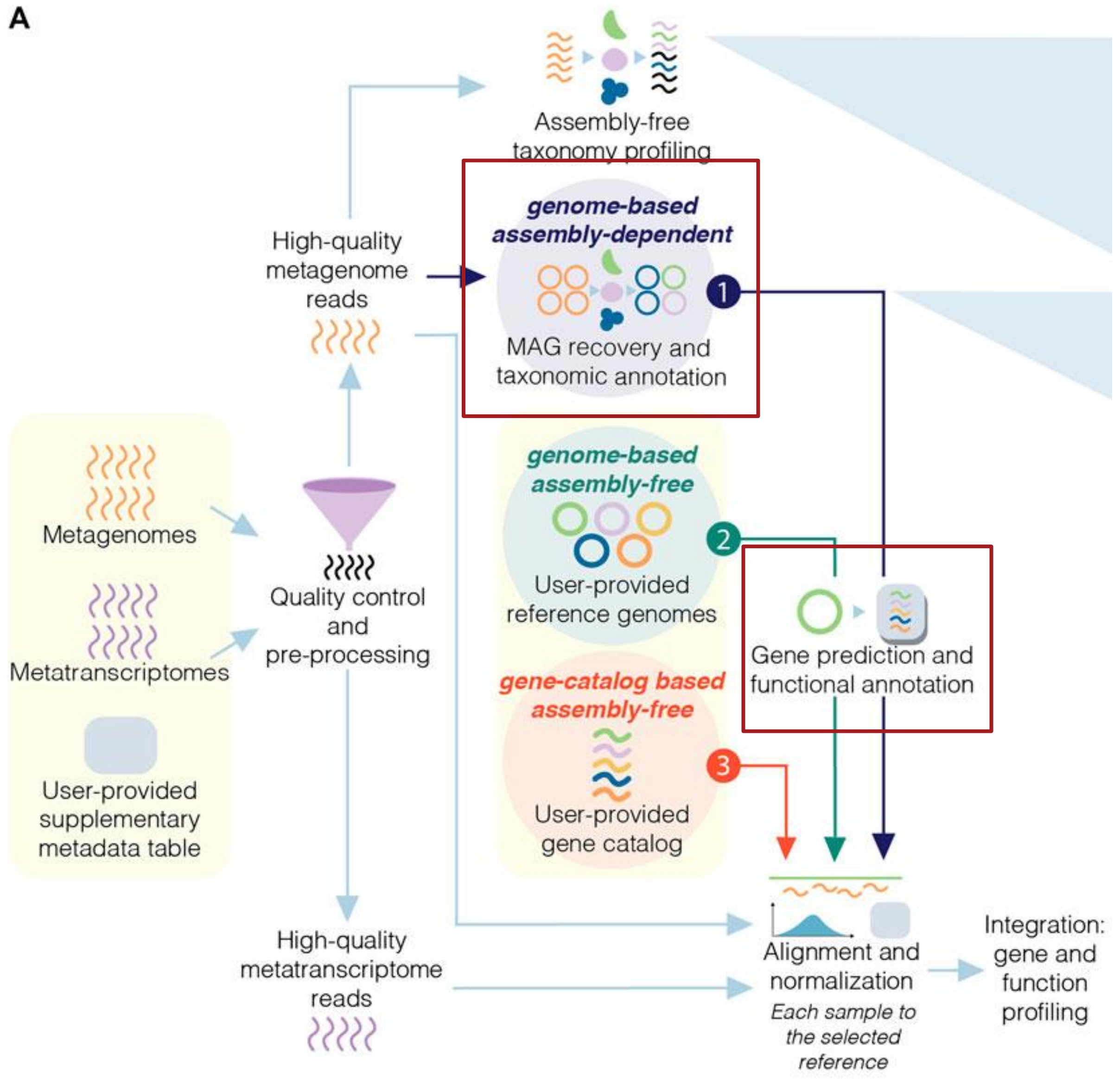
Recovery of annotated genomes from shotgun metagenomics data

PIG-PARADIGM workshop, 2024-04-29

MIntO

- A modular and scalable pipeline for microbiome shotgun metagenomics and meta-transcriptomics data integration
- https://github.com/arumugamlab/MIntO
- Snakemake workflow (python)
- Minimum hardware requirements:

Linux server with 16 CPUs and 64 GB RAM


A
Assembly-free taxonomy profiling
High-quality metagenome reads
genome-based assembly-dependent
1
MAG recovery and taxonomic annotation
Metagenomes
Metatranscriptomes
User-provided supplementary metadata table
Quality control and pre-processing
genome-based assembly-free
2
User-provided reference genomes
Gene prediction and functional annotation
gene-catalog based assembly-free
3
User-provided gene catalog
High-quality metatranscriptome reads
Alignment and normalization
Each sample to the selected reference
Integration: gene and function profiling

Dataset

- Also used in https://github.com/arumugamlab/MIntO/wiki/MIntO-Tutorial
- 10 samples from the Inflammatory Bowel Disease Multi'omics Database (IBDMDB) from two participants with Crohn's disease (CD1 and CD2)
- /home/ucdavis/tutorial/metaG

Part 1: Recovery of high quality genomes from shotgun metagenomics data

Overview of steps in the workflow:

- Assembly of sequencing reads into contiguous sequences
- Alignment of sequencing reads to contigs for coverage depth calculation
- Binning (grouping) of contigs to represent metagenome-assembled genomes (MAGs)
- Choosing unique MAGs with the highest quality
- Taxonomical annotation

Run setup commands and assembly.smk

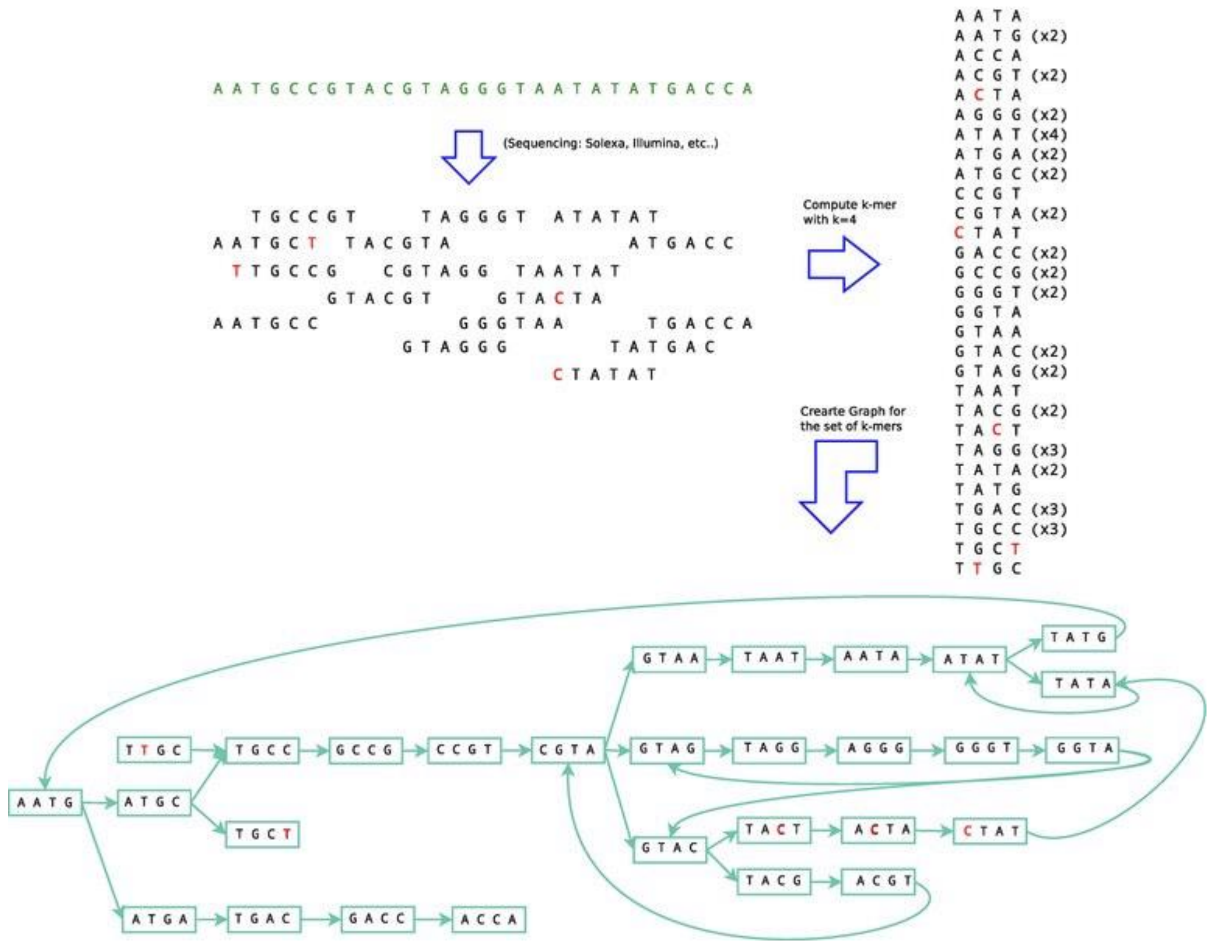
```
snakemake $SNAKE_PARAMS \
--snakefile $MINTO_DIR/smk/assembly.smk \
--configfile $PROJ_DIR/metaG/assembly.yaml -np
```

Assembly of sequencing reads into contiguous sequences

Gain longer, continuous sequences by connecting short reads via their overlaps using graphs

- Input: trimmed and error-corrected paired-end reads
  format: *fastq*

- Output: contigs (and scaffolds)
  format: *fasta*



Assembly of sequencing reads into contiguous sequences

- Software:
  - Individual samples: SPAdes *de novo* assembler in meta mode

```
spades.py --meta --only-assembler
-1 {input.fwd} -2 {input.rev} -t {threads} -m {resources.mem} -o {params.kmer_dir} \
--tmp-dir tmp --phred-offset auto -k 21,33,55,77,99
```

  - Co-assembly (multiple samples): MEGAHIT *de novo* assembler

```
megahit -1 {params.fwd_reads} -2 {params.rev_reads} -t {threads} -m
{resources.mem_bytes} --out-dir assembly {params.asm_params} --presets meta-large
```

Check output from assembly.smk

```
ll $PROJ_DIR/metaG/7-assembly/
```

Run binning_preparation.smk

```
snakemake $SNAKE_PARAMS \
--snakefile $MINTO_DIR/smk/binning_preparation.smk \
--configfile $PROJ_DIR/metaG/assembly.yaml
```

Alignment of sequencing reads to contigs for coverage depth calculation

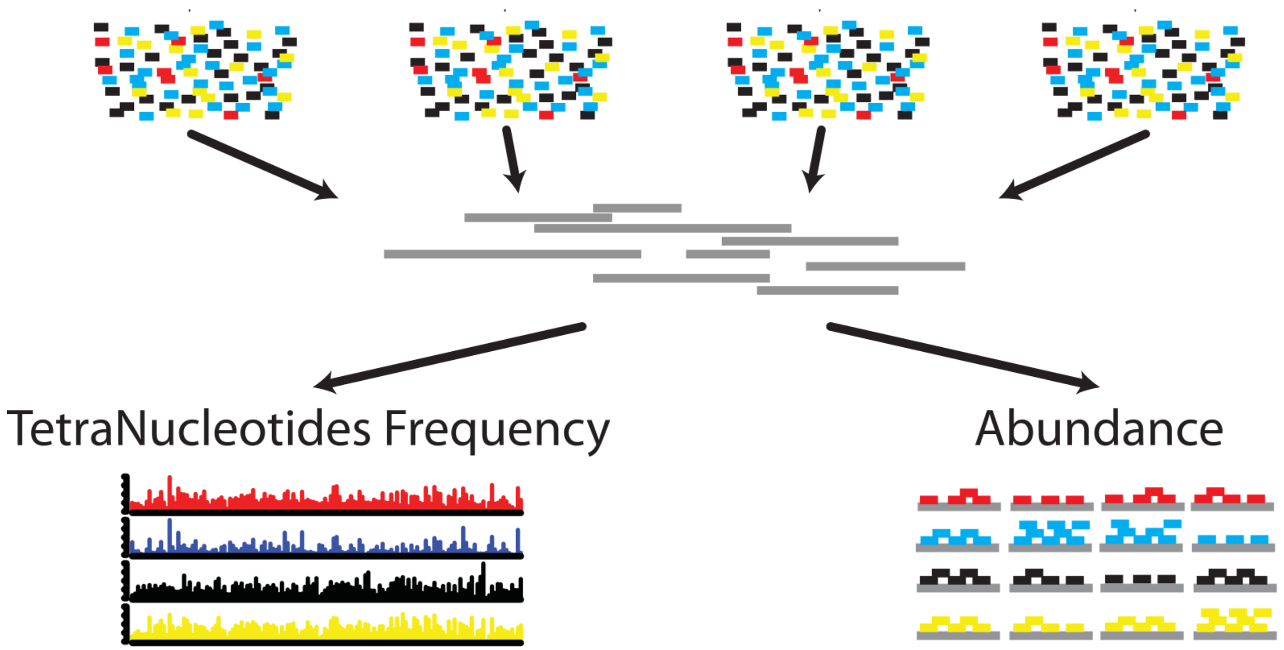
Co-abundance binning requires information about how many reads could be aligned from each sample to each assembly

- Input: trimmed and error-corrected paired-end reads
  contigs

- Output: depth files

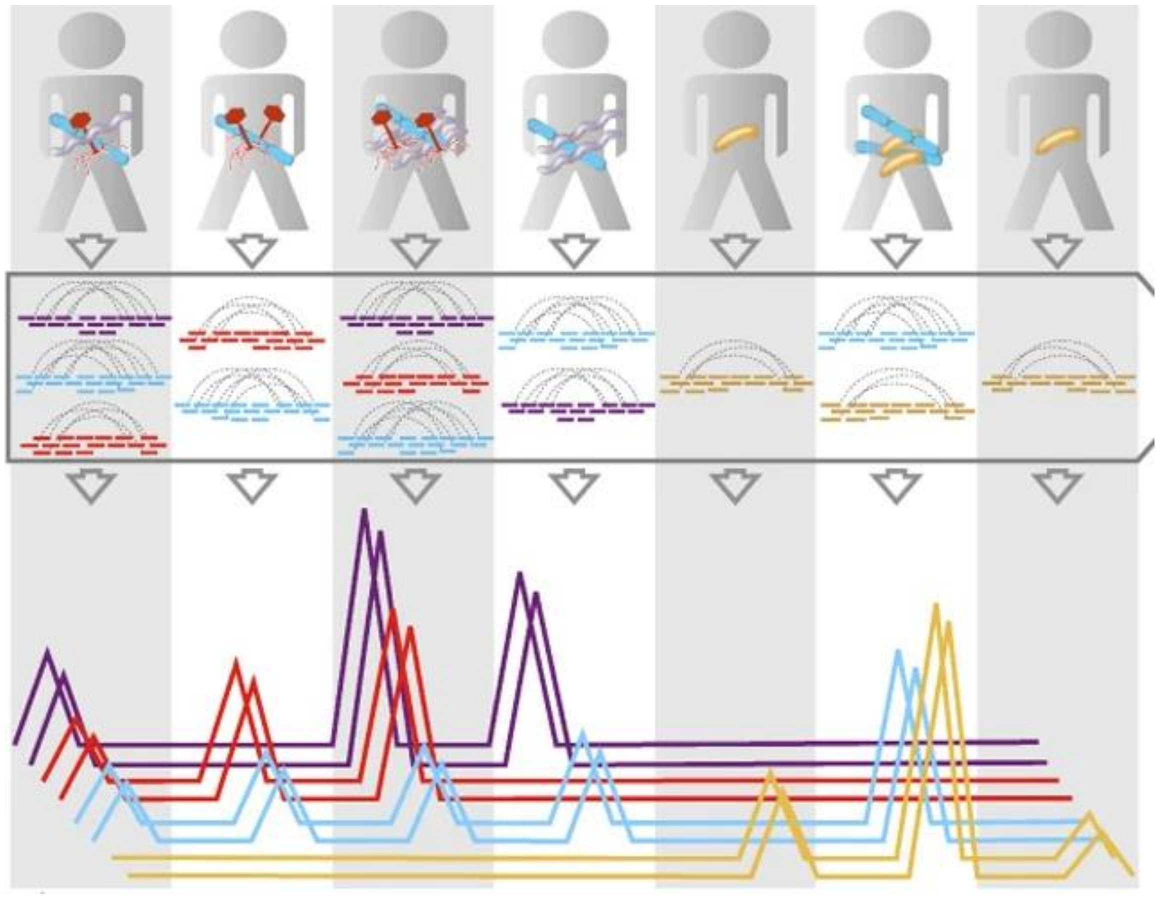
Alignment of sequencing reads to contigs for coverage depth calculation

- All samples' reads are aligned to all assemblies



Alignment of sequencing reads to contigs for coverage depth calculation

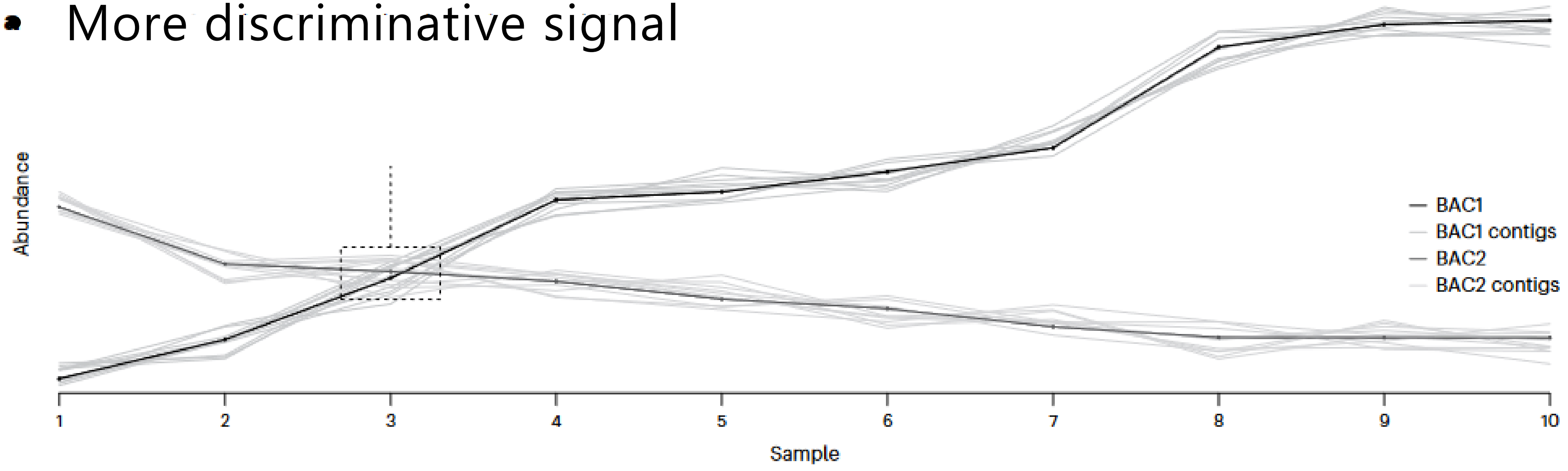
- More discriminative signal



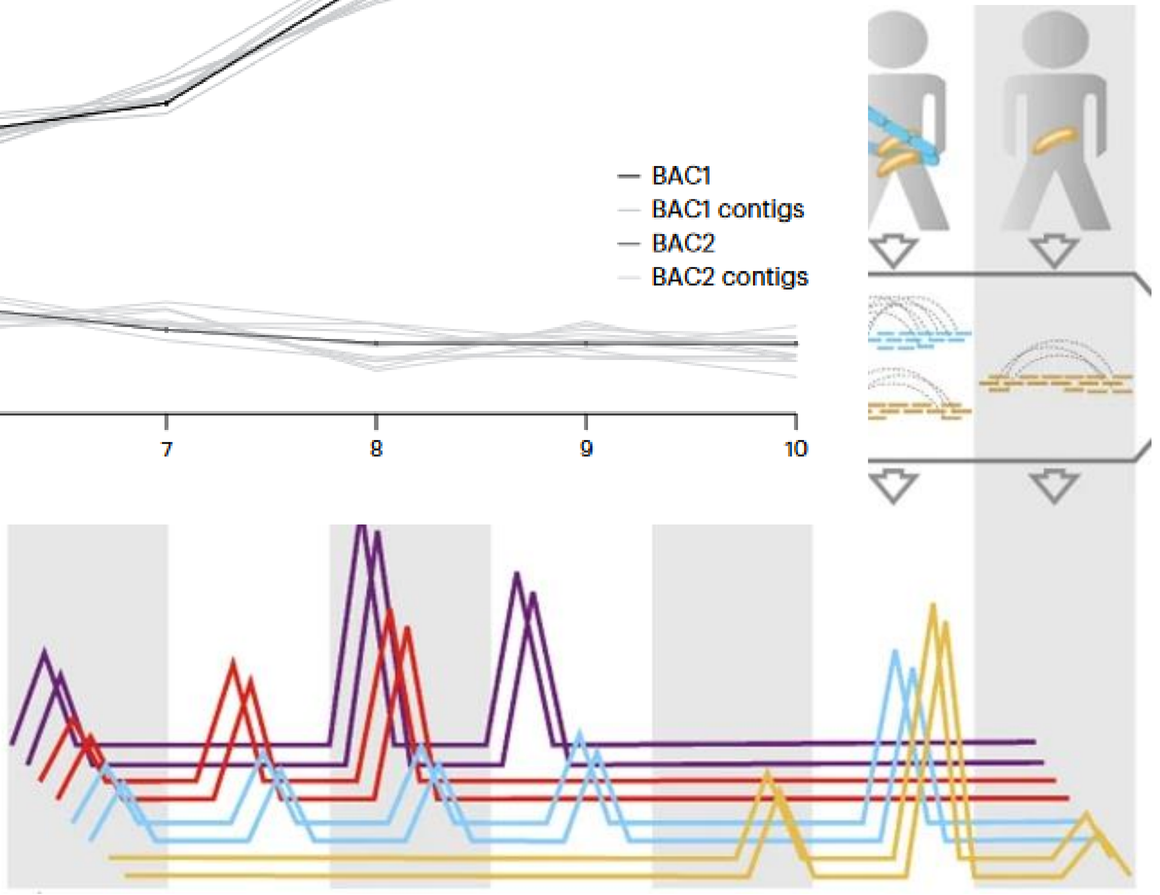
Nielsen 2014, Nat. Biot., 10.1038/nbt.2939

Alignment of sequencing reads to contigs for coverage depth calculation

- More discriminative signal



Watson and MAttock 2023, Nat. Meth, 10.1038/s41592-023-01934-8



Nielsen 2014, Nat. Biot., 10.1038/nbt.2939

Alignment of sequencing reads to contigs for coverage depth calculation

- Software:
  - Alignment: bwa-mem2
  - BAM file filtering, indexing and sorting by coordinate: samtools
  - Depth calculation and normalization: coverM

```
bwa-mem2 mem -P -a -t {params.map_threads} $db_name {input.fwd} {input.rev} \
| msamtools filter -buS -p 95 -l 45 - \
| samtools sort -m {resources.sort_mem}G --threads {params.sort_threads} - > {wildcards.illumina}.bam
coverm contig --methods metabat --trim-min 10 --trim-max 90 \
--min-read-percent-identity 95 --threads {params.coverm_threads} --output-file sorted.depth --bam-files
{wildcards.illumina}.bam
```

Check output from binning_preparation.smk

```
ll $PROJ_DIR/metaG/8-1-binning/
```

Run mags_generation.smk

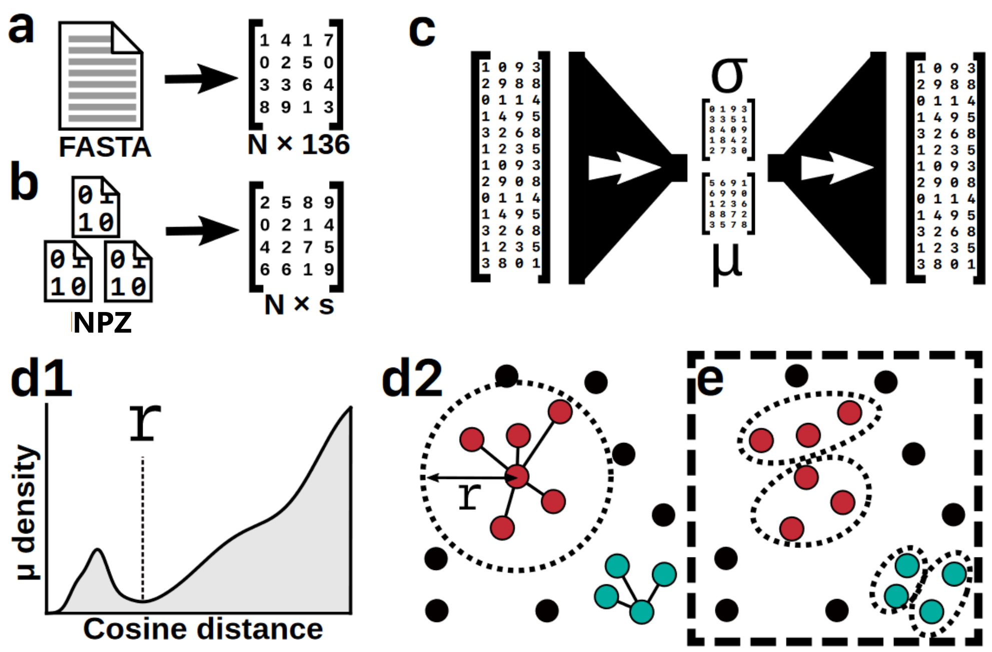
```
snakemake $SNAKE_PARAMS \
--snakefile $MINTO_DIR/smk/mags_generation.smk \
--configfile $PROJ_DIR/metaG/mags_generation.yaml
```

Binning of contigs to represent MAGs

Binning by integrating tetra-nucleotide frequency and abundance data, and clustering in a latent space (reduced dimensions) using variational-autoencoders

- Input: depth files (npz) and contigs (fasta)

- Output: MAGs (fasta)



Nissen et al 2021, Nat. Biot., 10.1038/s41587-020-00777-4

Binning of contigs to represent MAGs

- Software:
  - VAMB (vae384) binner
  - Custom script to form separate MAGs based on contigs originating assembly

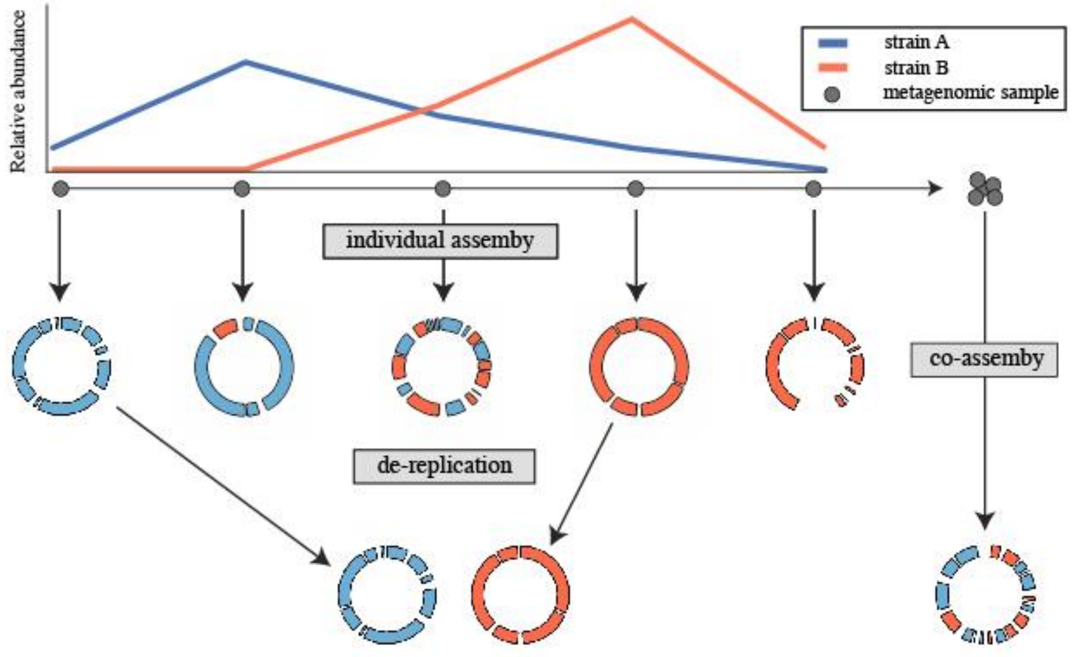
```
vamb --fasta {input.contigs_file} \
                --rpkm {input.rpkm_file} \
                --seed 1234 \
                -p {threads} \
                --cuda \
                --outdir out \
                --model vae \
                -l 24 \
                -n 384 384
```

Choosing unique MAGs with the highest quality

Creating a genome catalogue by selecting HQ MAGs with >90% completeness and <5% contamination, that are representing unique strains

- Input: MAGs from all assemblies

- Output: unique MAGs



https://drep.readthedocs.io/en/latest/overview.html

Choosing unique MAGs with the highest quality

- Software:
  - Quality estimation: CheckM2

```
checkm2 predict --quiet --database_path {params.checkm_db} -x fna \
--remove_intermediates --threads {threads} --input {input} --tmpdir tmp -o out
```

  - De-replication: coverM

```
coverm cluster --genome-fasta-directory {params.HQ_folder} \
--checkm2-quality-report {input.checkm_total} -x fna --cluster-method fastani \
--ani 99 --fragment-length 2500 --min-aligned-fraction 30 \
--output-cluster-definition {output.cluster_tsv} --threads {threads} \
--precluster-method finch --precluster-ani 93
```

Check output from mags_generation.smk

```
ll $PROJ_DIR/metaG/8-1-binning/mags_generation_pipeline
ll $PROJ_DIR/metaG/8-1-
binning/mags_generation_pipeline/unique_genomes
```

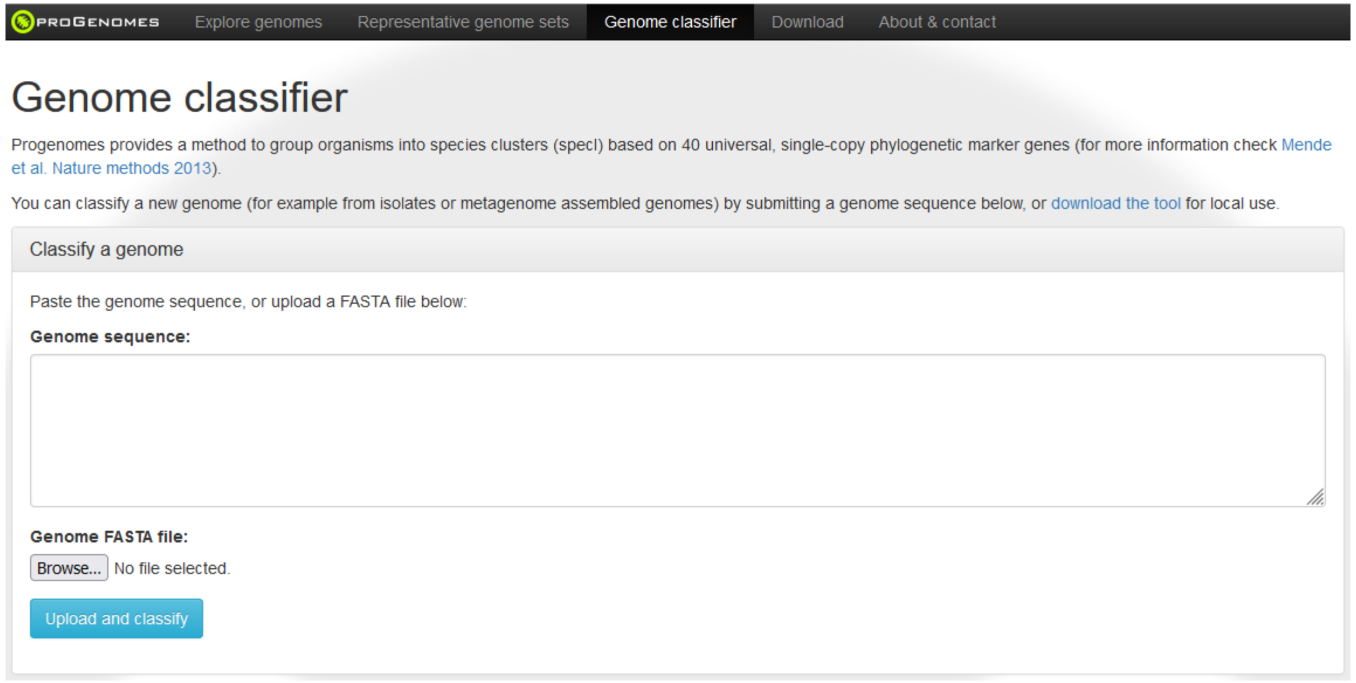
Taxonomical annotation of genomes

Usually, the genetically closest taxonomical label is assigned, as not all MAGs are expected to have representatives in the taxonomical databases.

- Input: unique MAGs (fasta)
- Output: taxonomic lineage

Taxonomical annotation of genomes

- Software:
  - GTDB-tk and/or PhyloPhlAn in MIntO
  - ProGenomes3 for us: https://progenomes.embl.de/classifier.cgi
    - Pick one genome and download it from the ucdavis server
    - Submit it to ProGenomes
    - Wait patiently



Genome classifier

Progenomes provides a method to group organisms into species clusters (specI) based on 40 universal, single-copy phylogenetic marker genes (for more information check Mende et al. Nature methods 2013).

You can classify a new genome (for example from isolates or metagenome assembled genomes) by submitting a genome sequence below, or download the tool for local use.

Classify a genome

Paste the genome sequence, or upload a FASTA file below:

Genome sequence:

Genome FASTA file:

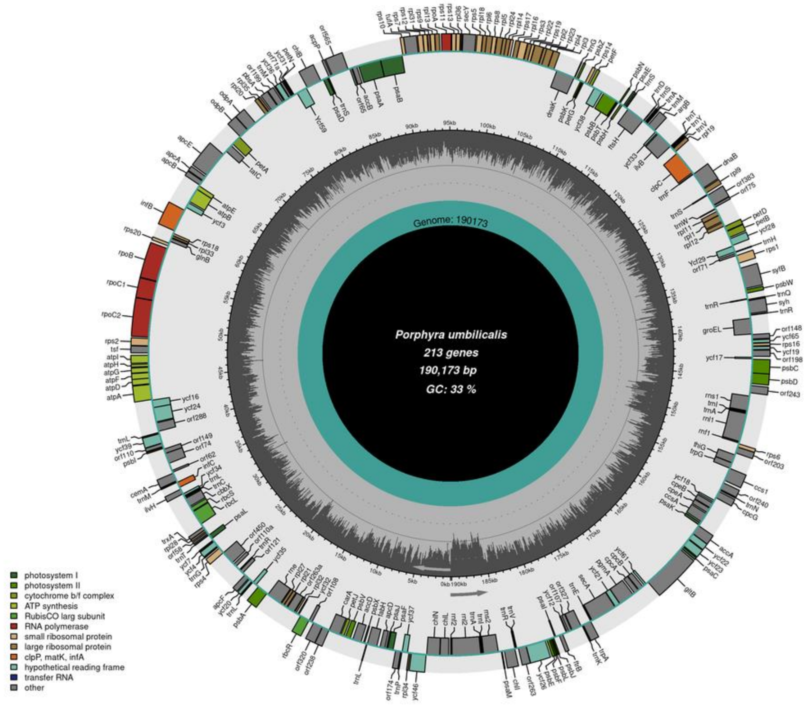
Browse... No file selected.

Upload and classify

Part 2: Functional annotation of genomes

Overview of steps in the workflow:

- Prediction of open reading frames (ORFs), rRNA, tRNA, etc. from the sequence of the genome and translation of coding sequences to protein sequences
- Assignment of functional annotations or orthologous groups based on homology



APashkov, Wikimedia 2023

Run gene_annotation.smk

```
snakemake $SNAKE_PARAMS \
--snakefile $MINTO_DIR/smk/gene_annotation.smk \
--configfile $PROJ_DIR/metaG/mapping.yaml
```

Prediction of ORFs and translation of CDSs to protein sequences

Functional annotation typically relies on homology to predict gene products. Functional orthologs are predicted from protein sequences.

- Input: unique MAGs (fasta)
- Output: ORFs' nucleotide sequences (fna)
  amino acid sequences (faa)
  genome annotation files (gff3, bed)

Prediction of ORFs and translation of CDSs to protein sequences

- Software:
  - Prokka
    - CDS: Prodigal
    - tRNA: Aragorn
    - rRNA: Barrnap
    - Protein families: HMMER3

```
prokka --outdir $(dirname {output.fna}) --prefix {wildcards.genome} --locustag
$locus_tag --addgenes --cdsrnaolap --cpus {threads} --centre X --compliant {input}
```

Assignment of functional annotations or orthologous groups based on homology

Simple example of assigning KEGG orthologous groups (KOs) for the gene products.

- Input: amino acid sequences (faa)
- Output: text file with KO numbers (https://www.genome.jp/kegg/ko.html)

Assignment of functional annotations or orthologous groups based on homology

- Software:
  - Kofamscan: assignment of KOs

```
exec_annotation -k {input.ko_list} -p {input.prok_hal} --tmp-dir tmp
--create-alignment -f mapper-one-line --cpu {threads} -o kofam_mapper.txt
{input.fasta_subset}
```

  - Custom script: add related modules and pathways

```
{script_dir}/kofam_hits.pl --pathway-map {input.pathway_map} --module-map
{input.module_map} kofam_mapper.txt > {output}
```

Assignment of functional annotations or orthologous groups based on homology

- Other software:
  - dbCAN: CAZyme annotation (https://github.com/linnabrown/run_dbcan)
  - eggNOG-mapper: Pfam, KEGG, OG, EC numbers, BIGG reactions, CAZyme, etc. (http://eggnog-mapper.embl.de/)

Check output from gene_annotation.smk

```
ll $PROJ_DIR/DB/9-MAG-genes-post-analysis/annot
```

That’s all for the commentary

- You can continue the exercise, if there is time until dinner.