Gene Prediction, Profiling & Annotation

A pipeline of the Vonaesch Lab for the curnagl cluster

Simon Yersin

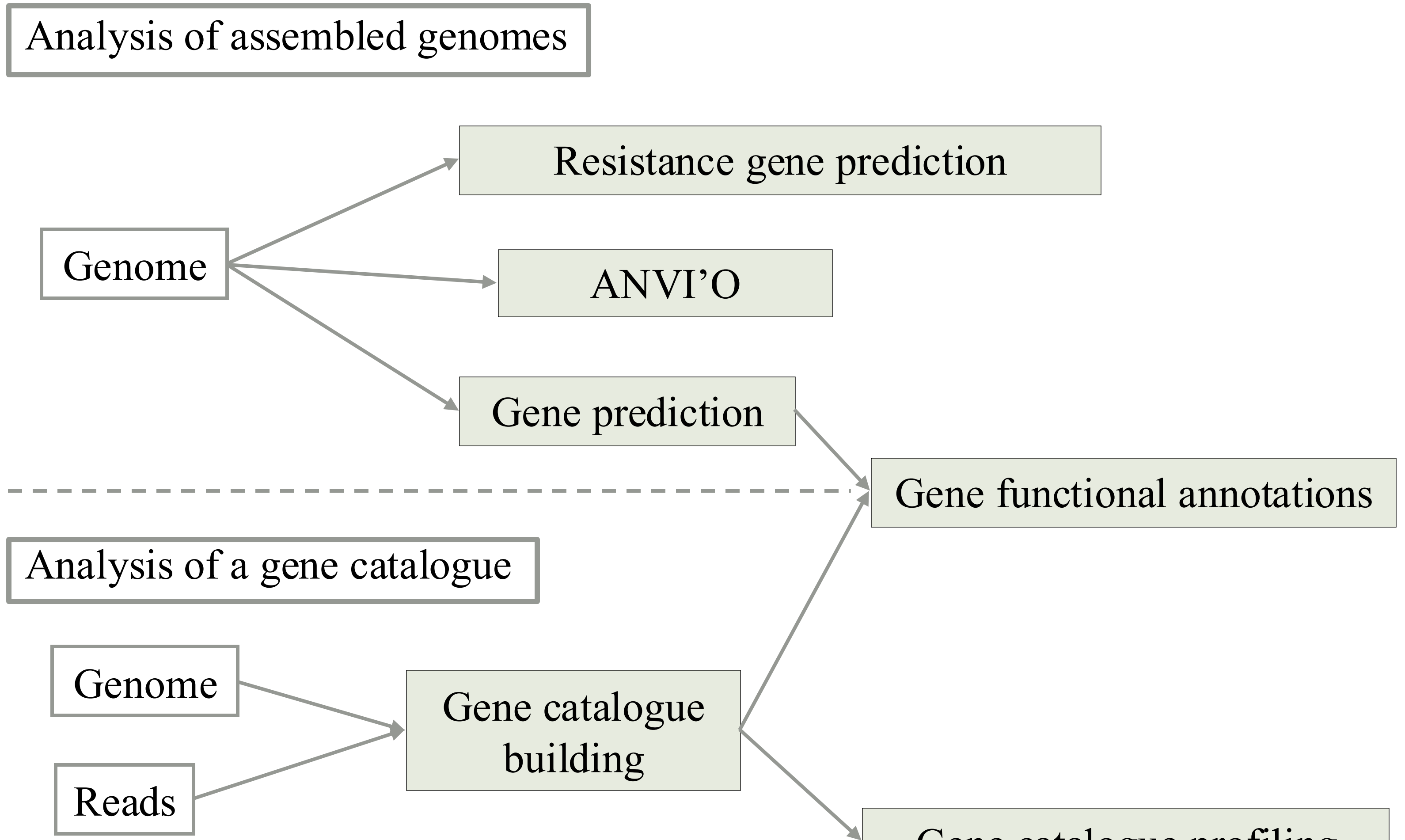
With the help of the Sunagawa Lab, the Engel Lab, Julian Garneau and the NCCR Methods in Microbiomics

Gene Prediction & Annotation

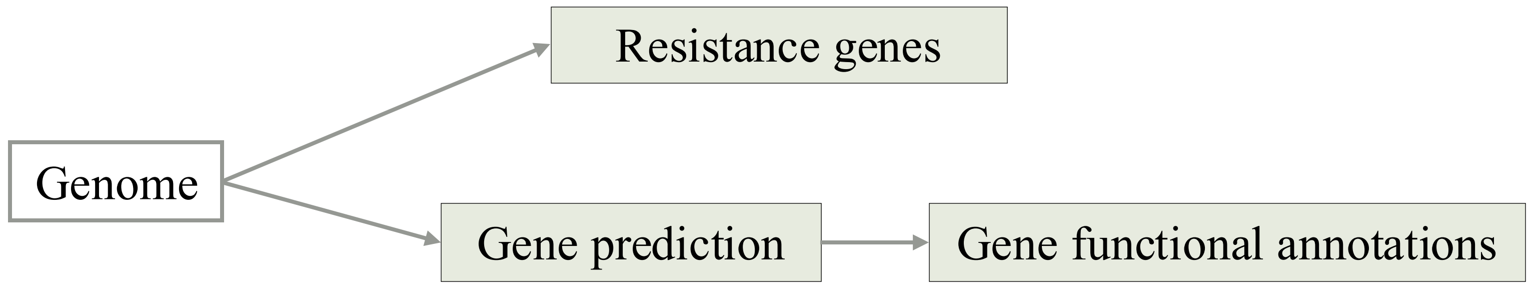
Downstream analysis can include the prediction and annotation of the gene in our dataset of interest. It can be separated in two categories:

- Analysis of assembled genomes (contigs or scaffolds)
- Analysis of a gene catalogue

Other downstream analysis can include: strain-diversity profiling, phylogeny and other type of analysis


Analysis of assembled genomes
Genome
Resistance gene prediction
ANVI'O
Gene prediction
Gene functional annotations
Analysis of a gene catalogue
Genome
Reads
Gene catalogue building
Gene catalogue profiling

Analysis of assembled genomes



Resistance genes in genomes (i.e. MAGs) can be profiled (i.e. annotated) directly without passing through gene prediction (i.e. prodigal) using either RGI or ABRicate.

For genomes, the gene prediction can be done with Prodigal, which will produce a .faa file for each genome. This file can be then annotated.

Resistance gene prediction in Genomes with RGI

- Resistance Gene Identifier (RGI) from the Comprehensive Antibiotic Resistance Database (CARD, a bioinformatic database of resistance genes, their products and associated phenotypes) can be used to predict resistome from protein and nucleotide data based on homology and SNP models.
- If DNA fasta sequences are submitted, RGI first predicts compete ORFs using Prodigal and analyzes the predicted protein sequences. This includes a secondary correction by RGI if Prodigal undercalls the correct start codon to ensure complete AMR genes are predicted. However, if Prodigal fails to predict an AMR ORF, RGI will produce a false negative result. If protein fasta sequences are submitted, RGI skips ORF prediction and uses the protein sequence directly.
- Short contigs, small plasmids, low quality assemblies, or merged metagenomic reads should be analysed using Prodigal's algorithms for low quality/coverage assemblies (contigs<20'000bp) and inclusion of partial gene prediction: parameter --low_quality
- The RGI analyses genomes or proteome sequences under a Perfect, Strict or Loos paradigm.
  - The Perfect algorithm is most often applied to clinical surveillance as it detects perfect matches to the curated reference sequences in CARD.
  - The Strict algorithm detects previously unknown variants of known AMR genes, including secondary screen for key mutations, using detection models with CARD's curated similarity cut-offs to ensure the detected variant is likely a functional AMR gene.
  - The Loose algorithm works outside of the detection model cut-offs to provide detection of new, emergent threats and more distant homologs of AMR genes, but will also catalog homologous sequences and spurious partial matches that may not have a role in AMR.
- Genomes, Genome assemblies, Metagenomic contig: RGI main
- Predict the resistome in your genomes, genome assemblies or metagenomic contig using:

rgi_main.sh

Resistance gene prediction with ABRicate

- ABRicate: Mass screening of contigs for antimicrobial resistance or virulence genes. It comes bundled with multiple databases: NCBI, CARD, ARG-ANNOT, Resfinder, MEGARES, EcOH, PlasmidFinder, Ecoli_VF and VFDB. ABRicate only supports contigs (not fastq reads) and only detects acquired resistance genes, not point mutations. It uses a DNA sequenced database, not proteins.
- Profile the resistance in the MAGs using:
  - abricate.sh
  - Select the database of interest
  - Results can be merged with abricate summary

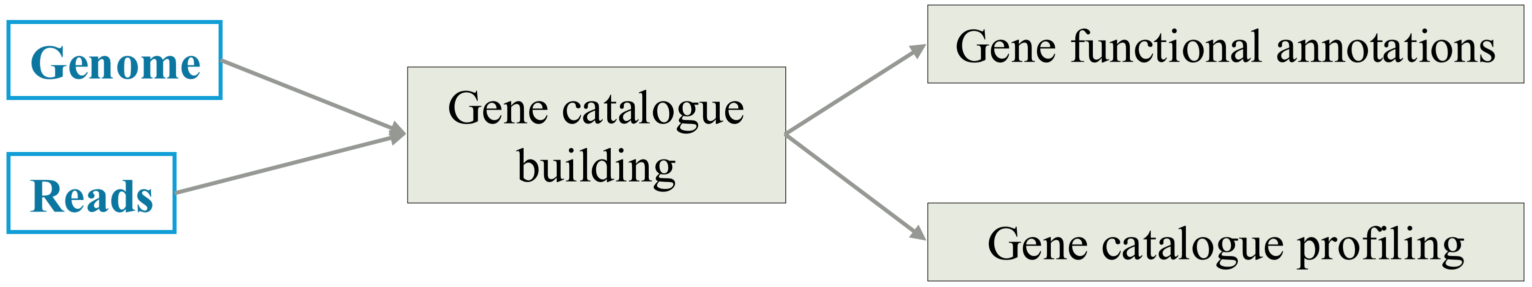
Resistance gene annotations

- https://www.nature.com/articles/s41598-021-91456-0
- metaMLST: https://github.com/SegataLab/metamlst

Gene Catalogue

Gene Catalogue

- Gene catalogue generation and profiling (i.e. gene abundance estimation) can provide important insights into the community's structure, diversity and functional potential. This analysis could also identify relationships between genetic composition and environmental factors, as well as disease associations.
- The creation of a *de novo* gene catalogue is performed in 3 main steps:
  - Building (combine assembled scaffolds and MAGs)
  - Profiling
  - Annotating



Gene catalogue building

Building the gene catalogue is done in three steps:

- **Gene calling**: We use prodigal to extract protein-coding genes from metagenomic assemblies (using scaffolds >= 500 bp as input) and MAGs. Prodigal has different gene prediction modes with single genome mode as default. To run prodigal on metagenomic data, we add the -p meta option while for MAGs we use the –p single option. This will produce a fasta file with amino acid sequences (.faa), nucleotide sequences (.fna) for each gene, as well as an annotation file (.gff).
- **Gene concatenation**: At this point gene-nucleotide sequences from all samples are concatenated together and contain duplicated.
- **Clustering:** gene-nucleotide sequences are clustered and representative sequence obtained using MMseqs2.

Gene prediction with Prodigal

- Prodigal is a protein-coding gene prediction software tool for bacterial and archaeal genomes. It handles draft genomes and metagenomes. Prodigal does not provide functional annotations, it predicts the coding regions but does not assign functions to them. Prodigal will produce a fasta file with amino acid sequences (.faa), nucleotide sequences (.fna) for each gene, as well as an annotation file (.gff).
- Gene calling is first done on the scaffold with minimum 500bp in length to include genes that would not be present in the final MAGs due to the binning methods and QC filtering. This way we profile genes that failed to assemble in a larger scaffold or failed to be binned in a MAG.
- Run prodigal on the min500 scaffold with the –p meta option using:
  - 1_prodigal_scaffolds.sh
- Run prodigal on the MAGs with the –p single option using:
  - 2_prodigal_mags.sh
- These scripts include a sequence headers modification in the resulting prodigal file. For scaffolds it add METAG and for mags it add MAG# in the sequence header to make sure each are unique

Gene catalogue concatenation

- Concatenate gene-nucleotide sequences from all samples together using:
  - mkdir gene_catalog
  - cat ./*.faa > gene_catalog/gene_catalog_all.faa
  - cat ./*.fna > gene_catalog/gene_catalog_all.fna
- Control there are no issue with the concatenation, all sequences are on their own line
  - grep -n '[^>]>\S' Afribiota_gene_catalog_all.fna
- Option 2: use

  3_merge_catalog.sh

- Check if sequence headers are unique (show duplicated):

```
grep '^>' gene_catalog_all.fna | sort | uniq -d
```

Gene clustering

- Genes are then clustered to obtain one representative gene for each cluster using MMseqs2. MMseq2 (Many-against-Many sequence searching) is a software suite to search and cluster huge protein and nucleotide sequence sets.
- This is done in 3 steps:

1. Create db: convert the fasta database into a the MMseqs2 database format

   4_mmseqs_db.sh

2. Cluster: run the clustering of the database, it returns the database files: _clu and _clu.index

   5_mmseqs_cluster.sh

3. Create tsv: generate a tsv formatted output file from the previous output file

   6_mmseqs_tsv.sh

- Save the resulting tsv file which contain the representative sequences for the cluster and which sequences are part of this cluster.

Info: MMseqs2 output files

Main database files:

- mmseqs.db input sequence database in MMseqs2 format (from fna.gz)
- mmseqs.db_h.* is the header database, storing the sequence identifiers
- mmseqs.db.index index of the main datbase
- mmseqs.db.dbtype specifies the database type (nucleotide vs protein)

Clustering result files:

- mmseqs.db.cluster main clustering result in MMseqs2 binary format, each sequence is linked to its representative sequence ID
- mmseqs.db.cluster.* are the split parts of the .cluster file, created during the split memory limit process, together they compose the full .cluster result
- mmseqs.db.cluster.dbtype, .index similar to inout db for the cluster db

Readable clustering output:

- **mmseqs.db.tsv human readable tsv file mapping the clustered sequences to their representative.**
- The format is: rep_seq_id		member_seq_id

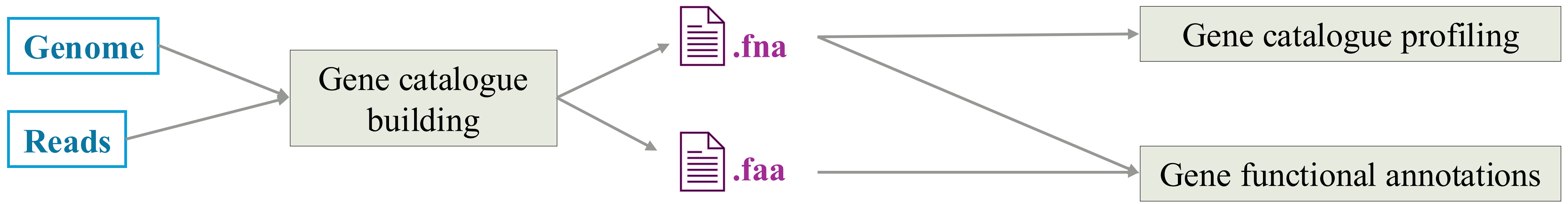
Log file: mmseqs.db.log

Gene clustering

- Extract representative sequence headers from the tsv file:
  - 7_headers.sh
- Filter the gene catalogue in protein (.faa) and nucleotide (.fna) using the header file and seqtk:
  - 8_fna_seqtk.sh
  - 9_faa_seqtk.sh
- Save the dereplicated gene catalogues in nucleotide (.fna) and protein space (.faa).
- Building of the gene catalogue is done!

Gene Catalogue

- The gene catalogue building steps produces two files:
  - The gene catalogue in nucleotides (.fna)
  - The gene catalogue in proteins (.faa)
- These files can be used for the next steps:
  - Profiling: quantify gene abundance in the samples
  - Annotating: annotate the genes
- These steps are done separately from each other



Gene catalogue profiling

Profiling means looking at the gene abundance in each sample

Profiling the gene catalogue is done in three steps:

- **Read alignment**: Once dereplicated, (cleaned) sequencing reads are mapped back to the gene catalog using BWA aligner. Note that forward and reverse reads are mapped separately and are then filtered and merged in a later step.
- **Filtering the alignment files**: To make sure that quantification of gene abundance relies only on high confidence alignments, the alignment files are first filtered to only include alignments with length > 45 nt and percent identity > 95%.
- **Counting gene abundance:** This step counts the number of reads aligned to each gene for each of the samples and normalize it with the fetched marker genes.

Align & Merge alignments

- First index the dereplicated and clustered gene catalogue file using bwa index using:
  - 5_index_catalog.sh
- Align the sequencing reads on the indexed gene catalogue using bwa mem and samtools, forward and reverse are done separately
  - 6_align.sh
- Then, SushiCounter is used for merging of the bam alignment files using the mergebam option:
  - 7_mergebam.sh

Counting gene abundance

- SushiCounter counter is used for counting gene abundance for each samples, producing instert.count file for each sample. This step includes the computation of the total bases and insert counts from the metagenomic fastq files using seqkit. These information is used to normalise the gene count by the total bases and insert count when provided to SushiCounter with the –ti and –tb options.
  - 📜8_counter.sh
- Then, profiles are merged using SushiCounter mergeprofiles:
  - 📜9_mergeprofiles.sh
- This step produces five files:
  - horizontal coverage: hcov.profile
  - Inserts: insertcounts.profile
  - **Normalised inserts: insertcounts.lengthnorm.profile (used for next step)**
  - Bases: basecounts.profile
  - Normalised bases: basecounts.lengthnorm.profile

Cell count normalization

- After normalisation by total base and insert count done automatically by SushiCounter, the gene count is normalized by the median marker genes counts. The 40 universal marker genes are first extracted using fetchMGs.pl (https://github.com/motu-tool/fetchMGs.pl), currently installed on the /work:
  - 10_fetchmgs.sh
- The normalization is done using the python script normalise_by_motus_v3.fmg.py
  - 11_normalize.sh

  Normalization allow to compare genes within samples and between samples
- The resulting file is the normalized abundance of the representative genes profiled in the samples!

Gene annotations

Gene annotations

The previous steps produced the gene catalogue file (.faa) after dereplication and clustering (building steps) or, for a genome, a file (.faa) with the predicted genes. Now, we would like to annotate the genes with functions. Functional annotations relies on various databases (such as KEGG or CAZymes) and, depending on our interests, the following annotations can be obtained:

- EggNOG: orthology relationships, functional annotations and gene evolutionary histories, using eggnog_mapper https://github.com/eggnogdb/eggnog-mapper
- **dbCAN2**: carbohydrate-active enzyme and substrate annotation, using run_dbcan https://github.com/linnabrown/run_dbcan?tab=readme-ov-file
- KEGG Ortholgy using **KofamScan** https://github.com/takaram/kofam_scan
- Metabolic gene cluster using gutSMASH https://github.com/victoriapascal/gutsmash
- Secondary metabolite using antiSMASH https://github.com/antismash/antismash

Gene catalogue splitting

If the gene catalogue contains more than 1 millions genes, it is necessary to split it in parts of ~500'000 genes to increase processing speed by parallelization. The gene catalogue can be split using seqkit:

```
seqkit split gene_catalog_derep.faa -p 10
```

-p is the number of parts the file will be split into

-s is to split by number of sequences instead of parts

The same can be applied for fna files.

After running the annotation tools, the outputs can be merged.

KEGG annotations

- ➢ KofamScan is a gene function annotation tool based on KEGG Orthology and hidden Markov model. You need KOfam database to use this tool. Online version is available on https://www.genome.jp/tools/kofamkoala/ .
- ➢ KofamScan is installed in the /work in /syersin/KOFAM/ and contains the executable file in ./ kofam_scan-1.3.0 and the databases in ./database. It needs three modules: ruby/ , hmmer/ , and parallel/20220522
- ➢ No need to activate a conda environment! You can show the help with:

  ```
  /work/FAC/FBM/DMF/pvonaesc/vonasch_lab_general/syersin/KOFAM/kofam_scan-1.3.0
  /exec_annotation –h
  ```
- ➢ Run KofamScan, on the split gene catalogue:

  kofamscan.sh
- https://github.com/bjtully/BioData/blob/master/KEGGDecoder/README.md Kegg-decoder determine completeness of various metabolic pathways (more relevant for MAGs than gene catalogue)
- Can be run with keggdecoder.sh

CAZymes annotations

- CAZymes are carbohydrates active enzymes. To annotate which genes are CAZymes, we use dbCAN3 annotation tool run_dbcan for automated CAZyme annotation. Run well with array on the split gene catalogue. Annotate the CAZymes using:
  - cazymes.sh
- dbCAN3 relies on a CAZymes database loaded in: /work/FAC/FBM/DMF/pvonaesc/vonasch_lab_general/syersin/dbCAN/database
- From alexalmaida/genofan GitHub, cazymes.sh also includes: dbcan_simplify.py which simplify dbCAN3 output.
  - Or run: `python dbcan_simplify.py cazymes_overview.txt > cazymes_simplified.txt`

EggNOG annotations

- EggNOG-mapper is a tool for fast functional annotation of novel sequences. It uses precomputed Orthologous Groups (OGs) and phylogenies from the EggNOG database (http://eggnog5.embl.de) to transfer functional information from fine-grained orthologs only. Run well with array on the split gene catalogue. Annotate using EggNOG-mapper using:
  - eggnog_mapper.sh
- Parameters to define:
  - --itype proteins : type of sequence in the input
  - --output_dir where output file should be written
  - --scratch_dir write output files in a temporary scratch dir, move them to the final output dir when finished, cannot be the same than output_dir!
  - --temp_dir where temporary files are created
  - --data_dir path to database in the work directory /syersin/EGGNog/database

DRAM

DRAM (Distilled and Refined Annotation of Metabolism) is a tool for annotating metagenomic assembled genomes and VirSorter identified viral contigs. DRAM annotates MAGs and viral contigs using KEGG (if provided by the user), UniRef90, PFAM, dbCAN, RefSeq viral, VOGDB and the MEROPS peptidase database as well as custom user databases. DRAM is run in two stages. First an annotation step to assign database identifiers to gene, and then a distill step to curate these annotations into useful functional categories. Additionally, viral contigs are further analyzed during to identify potential AMGs. This is done via assigning an auxiliary score and flags representing the confidence that a gene is both metabolic and viral.

From a gene catalogue, the annotation can be done directly on the .faa file by specifying in distill the options --skip_trnas and --min_contig_size 0

Not working yet

Metabolic gene cluster annotations

- Anaerobic bacteria in the gut are responsible for the synthesis and transformation of diverse molecules involved in host-microbe and microbe-microbe interactions, which ultimately mediate host phenotypes. The pathways for the production of these molecules belong to specialized primary metabolism and the genes encoding them are often physically clustered in the genome, in regions also known as metabolic gene clusters (MGCs).
- gutSMASH is a tool that has been developed to systematically evaluate the metabolic potential of these bacteria by predicting both known and novel anaerobic MGCs from the gut microbiome. Altogether, gutSMASH provides a comprehensive toolkit to functionally characterize anaerobic bacterial genomes by not only predicting MGCs of known functions but also novel MGCs that may represent good candidates for further experimental characterization.
- Not installed

Secondary metabolite annotations

- antiSMASH allows the rapid genome-wide identification, annotation and analysis of secondary metabolite biosynthesis gene clusters in bacterial and fungal genomes. It integrates and cross-links with a large number of in silico secondary metabolite analysis tools.

- Not installed
- Struggle to install with conda