IMP3

Release 3.0.0

May 23, 2022

Installation

The Integrated Meta-Omics Pipeline, IMP, is a reproducible and modular pipeline for large-scale integrated analysis of coupled metagenomic and metatranscriptomic data. IMP3 incorporates read preprocessing, assembly (including optional iterative co-assembly of metagenomic and metatranscriptomic data), analysis of microbial community structure (taxonomy) and function (gene functions and abundances), binning, as well as summaries and visualizations.

IMP3 is implemented in Snakemake. It is user-friendly and can be easily configured to perform single-omic metagenomic or metatranscriptomic analyses, as well as single modules, e.g. only assembly or only binning.

IMP3 is being developed at the Luxembourg Centre for Systems Biomedicine and the Swammerdam Institute for Life Sciences at the University of Amsterdam.

CHAPTER 1

How to install IMP3

To install IMP3, you need conda.

1. Clone this repository to your disk:

```
git clone --recurse-submodules --single-branch --branch master https://git-r3lab.uni.
↪lu/IMP/imp3.git ./IMP3
```

Change into the IMP3 directory:

```
cd IMP3
```

At this point, you have all the scripts you need to run the workflow. You still need to download a number of databases, though (see point 3).

If you want to use the **comfortable IMP3 wrapper**, follow the points 4-8. If you don't want to use the wrapper, point 9 has a hint on the initialization of the conda environments.

2. Create the main `conda` environment:

If you don't have `conda`, follow these instructions first.

Create the main environment to be able to execute `snakemake` workflows.

```
conda env create -f requirements.yaml --prefix ./conda/snakemake_env
```

To run any `snakemake` workflow, you have to activate the created environment:

```
conda activate ./conda/snakemake_env
```

3. Download databases:

Note: if you already have some of these databases, you don't need to download them again. You need to link them into the directory that keeps all of the databases, though. You may also not need all of these databases, if you don't run the steps that use them.

There is an installation workflow (see directory `install/`) which will setup the requirements defined in a config file. A default configuration is provided in the file *config/config.install.yaml*.

Here is an example how to execute the setup:

```
# activate the snakemake environment
conda activate ./conda/snakemake_env
# dry-run
snakemake -s install/Snakefile --configfile config/config.install.yaml --cores 1 -rpn
# execute (use conda, more cores)
snakemake -s install/Snakefile --configfile config/config.install.yaml --cores 8 --
↪use-conda --conda-prefix ./conda -rp
conda deactivate
```

Please note that the cores, memory and runtime requirements depend on the used config file.

For reference, see below for more information on how the requirements have to be set up for the different workflow steps.

Step preprocessing:

- Trimmomatic adapters: adapters should saved in a subdirectory in your database directory called `adapters/` . You can find the latest version of Trimmomatic adapters here . .. We also supply such a directory (from Trimmomatic v.0.32) that you'd just need to unzip. You may also want to use different adapters, which you can just supply in FASTA format but make sure to use the format expected by Trimmomatic. .. You specify the adapter set in the config file.
- Genomes to filter your reads against: reference FASTA files (with extention `*.fa`) should be saved in a subdirectory in your database directory called `filtering/`. Usually, you want to remove phiX174 and host genome sequences.
- SortMeRNA databases: when processing metatranscriptomic reads, the rRNA reads have to be removed. These are filtered by SortMeRNA against the provided reference FASTA files. .. We supply these here . .. You need to unzip this into your database directory, or manually download the sequences from https://github.com/biocore/sortmerna/tree/master/data/rRNA_databases and put them into a subdirectory in your database directory called `sortmerna`.

Step analysis:

TODO: Mantis

Step taxonomy:

- Kraken2: to run taxonomic classification with Kraken2, a directory containing the required databases files has to be put (or linked) in your database directory. .. You will need a kraken2 database in your database directory (or a link to one). See here for a collection of Kraken2 databases. You can of course also use a custom database. .. You may also want to build your own. You just supply the name of the k2 database in the config file later, e.g.
- GTDB-tk: if you have performed **binning**, you need the GTDB database (file `gtdbtk_data.tar.gz`). .. Untar it it to a subdirectory called `GTDB_tk` within your database directory.

4. Adjust the file VARIABLE_CONFIG to your requirements (have a tab between the variable name and your setting):

- SNAKEMAKE_VIA_CONDA - set this to true, if you don't have snakemake in your path and want to install it via conda (recommended, because that way you have a current version). Leave empty, if you don't need an additional snakemake.
- LOADING_MODULES - insert a bash command to load modules, if you need them to run conda. Leave empty, if you don't need to load a module.
- SUBMIT_COMMAND - insert the bash command you'll usually use to submit a job to your cluster to run on a single cpu for the entirety of the run (a few hours to days). You can also include extra arguments, in particularly, if you need to bind jobs to one node (e.g. sbatch –nodelist=). You only need this whole setting, if you want to

have the snakemake top instance running in a submitted job. You alternatively have the option to run it on the frontend via tmux. Leave empty, if you want to use this version and have tmux installed.

- SCHEDULER - insert the name of the scheduler you want to use (currently *slurm* or *sge*). This determines the cluster config given to snakemake, e.g. the cluster config file for slurm is config/slurm.config.yaml or config/slurm_simple.config.yaml. Also check that the settings in this file is correct. If you have a different system, contact us and feel free to submit new scheduler files.
- MAX_THREADS - set this to the maximum number of cores you want to be using in a run. If you don't set this, the default will be 50. Users can override this setting at runtime.
- BIGMEM_CORES - set this to the maximum number of bigmem cores you want to be using in a run. If you don't set this, the default will be 5.
- NORMAL_MEM_EACH - set the size of the RAM of one core of your normal copute nodes (e.g. 8G). If you're not planning to use binny to submit to a cluster, you don't need to set this.
- BIGMEM_MEM_EACH - set the size of the RAM of one core of your bigmem (or highmem) compute nodes. If you're not planning to use binny to submit to a cluster or don't have separate bigmem nodes, you don't need to set this.
- NODENAME_VAR - if you submit your workflow to a cluster that requires binding to a fixed node, set this variable according to the environmental storing the nodename (e.g. SLURMD_NODENAME for most slurm systems)
- BIND_JOBS_TO_MAIN - if you submit your workflow to a cluster that requires binding to a fixed node, set this variable to true

5. Decide how you want to run IMP3, if you let it submit jobs to the cluster:

Only do one of the two:

- if you want to submit the process running snakemake to the cluster:

```
cp runscripts/runIMP3_submit.sh runIMP3
chmod 755 runIMP3
```

- if you want to keep the process running snakemake on the frontend using tmux:

```
cp runscripts/runIMP3_tmux.sh runIMP3
chmod 755 runIMP3
```

6. **optional, but highly recommended**: Install snakemake (and other dependencies) via conda:

If you want to use snakemake via conda (and you've set SNAKEMAKE_VIA_CONDA to true), install the environment, as recommended by Snakemake : Please note that the IMP3 pipeline is tested only using package versions specified in *requirements.yaml*. Using different versions than the specified ones might cause issues.

```
conda env create -f requirements.yaml --prefix $PWD/conda/snakemake_env
```

7. **optional, but highly recommended**: Set permissions / PATH:

IMP3 is meant to be used by multiple users. Set the permissions accordingly. I'd suggest:

- to have read access for all files for the users, **plus**:
- execution rights for the runIMP3 file and the .sh scripts in the subfolder submit_scripts
- read, write and execution rights for the conda subfolder
- to add the IMP3 directory to your path.
- It can also be useful to make the VARIABLE_CONFIG file not-writable, because you will always need it. The same goes for config.imp.yaml once you've set the paths to the databases you want to use (see below).

8. Initialize conda environments:

This run sets up the conda environments that will be usable by all users and will download a database:

```
./runIMP3 -i config/config.imp_init.yaml
```

This step will take several minutes to an hour.

9. Initialize the conda environments without wrapper:

This sets up the conda environments that will be usable by all users and will download more databases:

```
snakemake --cores 1 -s Snakefile --conda-create-envs-only --use-conda --conda-prefix␣
↪`pwd`/conda --configfile config/config.imp_init.yaml --local-cores 1
```

This step will take several minutes to an hour. I strongly suggest to **remove one line from the activation script** after the installation, namely the one reading: *R CMD javareconf > /dev/null 2>&1 || true*, because you don't need this line later and if two users run this at the same time it can cause trouble. You can do this by running:

```
sed -i "s/R CMD javareconf/#R CMD javareconf/" conda/*/etc/conda/activate.d/activate-
↪r-base.sh
```

CHAPTER 2

Running and configuring IMP3

2.1 Running IMP3

You can run IMP3 by calling Snakemake with the appropriate settings:

```
snakemake -s /path/to/IMP3/Snakefile --configfile /path/to/config.file.yaml --use-
↪conda --conda-prefix /path/to/IMP3/conda --cores corenumber
```

`Snakefile` defines your workflow that you want to execute.

`--configfile` specifies the config file for your sample. More info you can find *here*.

`--use-conda` enables Snakemake to use conda environments defined for your workflow.

`--conda-prefix` path to your IMP3 conda directory for the IMP3 conda environments.

`--cores` specifies the number of threads you want to maximally use at the same time in addition to the thread running the main `snakemake` command (but note *extra limitations* set in the config file).

All paths can be relative or absolute.

If you'd like to have a **report** which lists all IMP3 conda environments, rules and running stats, in addition to the visual outputs, run Snakemake with the `--report` option:

```
snakemake /path/to/IMP3/Snakefile --configfile config.file.yaml --use-conda --conda-
↪prefix /path/to/IMP3/conda --cores corenumber --report
```

If you want to use a batch submission system to run IMP3, add the appropriate arguments (more details *here*):

```
snakemake /path/to/IMP3/Snakefile --configfile config.file.yaml --use-conda --conda-
↪prefix /path/to/IMP3/conda --cores corenumber --cluster-config /path/to/IMP3/config/
↪cluster.config.yaml --cluster "{cluster.call} {cluster.runtime}{params.runtime}
↪{cluster.mem_per_cpu}{params.mem} {cluster.threads}{threads} {cluster.partition}
↪{cluster.nodes}"
```

2.2 Configuring IMP3 runs

IMP3 has a modular design and is highly configurable. Users can define the analysis steps IMP3 should perform, the data it should run on, the databases and tools that should be used and the settings of each step in a single **config file** in yaml format. This **config file** is provided to Snakemake as described *here*, using the `--configfile` argument.

This description guides through all sections and fields of the **config file**.

Some general notes on the **config file**:

- The **order of the items** in the config file is **arbitrary**.
- The config file does **not** need to contain all fields:
- only two fields are **required**: *raws* and an **output directory**.
- all other fields are optional and if emtpy the IMP3 default settings will be used.
- IMP3 will write the user-specified configuration including pre-defined fields to the output folder as `sample.config.yaml`.

2.2.1 Inputs, outputs and directories

The user has to tell IMP3 where the **raw or input files** are located. (The default is no input, and nothing will be done). Different *steps* in IMP3 require different input files. All input files are given as absolute paths. Input files can be:

- `Metagenomics` and/or `Metatranscriptomics` paired-end or single-end raw FASTQ files. Paired end read files are separated by `<space>`, or
- `Alignment_metagenomics` and/or `Alignment_metatranscriptomics`: MetaG and/or MetaT BAM files (requires `Contigs` file, required if the *Preprocessing* and *Assembly* steps are skipped).
- `Contigs` file in FASTA format (used for the mapping of the metaG und metaG raw files, required if the *Preprocessing* and *Assembly* steps are skipped).
- `LongReads` file in FASTQ (requires a `LongReadTech` description). `LongReads` are currently only supported with additional short **metaG** reads and with Spades assembler.
- `LongReadTech` description of used long read sequencing technology, currently `nanopore` or `pacbio`.
- `Gff` file based on the `Contigs` file (only required if the *Preprocessing*, *Assembly* and *Analysis* steps are skipped and only the step *Binning* should performed).

The `outputdir` (with full path) is a required field and defines the directory the IMP3 workflow output should be written to.

Users can give a **sample name** (`sample`, default value is `test`). Special characters should be avoided, punctuation will be converted to underscore. The **sample name** will be used throughout the IMP3 workflow, e.g. prepended to contig names. If the user-defined **sample name** is empty `sample: ""`, IMP3 will take the last two parts of the output path, concatenate them with an underscore and use the results as sample name.

A directory for **temporary files** can be given (`tmp_dir`, default name is `tmp`). If an absolute path is not given, the `tmp` directory will be placed into the output directory.

If IMP3 performs any of the steps that need *databases* (i.e. *Preprocessing* (filtering of reads against a host genome, filtering rRNA reads), *Analysis* (gene calling, functional annotation), or *Taxonomy* (kraken2), the user has to give the full path to a single directory with all the databases.

```
raws:
  Metagenomics: "/path/to/pair1 /path/to/pair2"
  Metatranscriptomics: "/path/to/pair1 /path/to/pair2"
  LongReads: "/path/to/longreads"
  LongReadTech: "nanopore|pacbio"
  Contigs: "/path/to/contigfile"
  Alignment_metagenomics: "/path/to/mg-bamfile"
  Alignment_metatranscriptomics: "/path/to/mt-bamfile"
  Gff: "/path/to/gfffile"

outputdir: "/path/to/output"
sample: <sampleid> (default: test)
tmp_dir: /path/to/tmp (default: tmp)
db_path: "/path/to/IMP_DBs"
```

2.2.2 IMP3 steps

IMP3 runs either the whole workflow or just some *steps*. The field `steps` lists the user-defined IMP3 analysis steps, separated by a space. The default is to run all steps.

The first step in IMP3 is *Preprocessing*, i.e. quality of reads, trimming and the removal of reads which map against a reference genome, most commonly a host genome). The default setting (`preprocessing_filter: true`), given that **metaG** or **metaT** reads are provided, will quality-trim reads and map them mapped against the human genome given as file `hg38.fa` in the database directory under `db_path`. The user can set `preprocessing_filtering` to `false` to skip this step, and the trimmed, unfiltered reads will directly go into the *Assembly* step.

If `steps` contains `summary`, IMP3 will summarize the results. The user can define which steps should be taken for summary in the field `summary_steps`. The default is to extract summary statistics (`stats`) on all performed steps and to visualize them (`vis`). If `summary_steps` is empty, IMP3 will not make summaries. Currently, the user can either choose `"stats"` or `"stats vis"`.

```
steps: "preprocessing assembly analysis binning taxonomy summary"
preprocessing_filtering: true
summary_steps: "stats vis"
```

2.2.3 Available hardware

Some IMP3 steps need high memory cpus depending on the size of the dataset and the databases used. IMP3 is able to run tasks on specific type of nodes. High memory demanding jobs are automatically scheduled to `big_mem` nodes. Depending on how much **RAM** on the local computer or compute cluster are available for the IMP3 run, the user can define the settings for **normal** and **bigmem** cpus:

- `normal_mem_per_core_gb` max. available RAM (in Gb) per core of **normal** compute cores, default 4 Gb
- `big_mem_total_gb` max. available total RAM (in Gb) of **bigmem** compute cores, default 1600 Gb
- `big_mem_cores` available number of **bigmem** compute cores, default 8
- `big_mem_per_core` max. available RAM (in Gb) per core of **bigmem** compute cores, default 200 Gb

The number of cores that are actually used at any point in time is determined by the `--cores` argument of the Snakemake *call*, rather than set here.

Note: *Defaults may be blatantly inappropriate for your system. Most steps will run with parameters that don't match your system, but some may cause errors (e.g. mapping, kraken, assembly).*

Currently you need to have both kinds of cores, if you run IMP3 in *cluster mode*, or you have to change the *cluster config file*.

```
mem:
  normal_mem_per_core_gb: 4
  big_mem_total_gb: 1600
  big_mem_cores: 8
  big_mem_per_core_gb: 200
```

2.2.4 Pre-processing: trimming

For the IMP3 *Preprocessing* step, IMP3 uses Trimmomatic for read trimming. The user can set all `trimmomatic` options (defaults given below).

If the reads originate from an Illumina Nextseq machine, we advise to set `nextseq` to true. This will remove trailing G's that the Nextseq systems add to reads that are shorter than the run cycles.

```
trimmomatic:
  adapter: "TruSeq3-PE"
  leading: 20
  minlen: 40
  palindrome_clip_threshold: 30
  simple_clip_threshold: 10
  trailing: 20
  seed_mismatch: 2
  window_size: 1
  window_quality: 3
  strictness: 0.5
  target_length: 40
nextseq: false
```

2.2.5 Pre-processing: filtering

The IMP3 *Preprocessing* step allows to map against a user-defined reference genome (e.g. host). The user can set the file name (minus suffix) of the reference genome (`filter`), which should be located or (soft-)linked to the *databases directory* (`db_path`) subfolder `filtering`, by default this is a FASTA file `hg38.fa` within the directory `db_path/filtering`.

IMP3 will remove rRNA reads from **metaT** reads. The user can decide which databases to use for this. The default is to use the databases supplied by SortMeRNA. These databases should be available in the *databases directory* in FASTA format.

```
filtering:
  filter: hg38
sortmerna:
  files:
    - rfam-5.8s-database-id98
    - silva-arc-16s-id95
    - silva-bac-16s-id90
    - silva-euk-18s-id95
    - rfam-5s-database-id98
    - silva-arc-23s-id98
    - silva-bac-23s-id98
    - silva-euk-28s-id98
```

2.2.6 Assembly

For the IMP3 *Assembly* step, the user has the choice between two assemblers (`assembler`): Megahit or MetaSpades. Both assemblers are available for **metaG assemblies**. If the input are **metaG** and **metaT** reads, the user can choose between a **hybrid** assembly using both types of reads (`hybrid: true`, default) or a purely **metaG-only** assembly (`hybrid: false`). **Hybrid metaG/metaT assemblies** are currently only possible with Megahit. MetaSpades is currently the only option to assemble **long and short metaG** reads together.

For all assemblies, the user can set the minimal (`mink`) and maximal (`maxk`) *k*-mer size and the intervals between (`step`) for the multi *k*-mer assemblies performed by both assemblers. IMP3 performs *iterative assembly* and the steps are merged using the CAP3 overlap assembler, which can be configured (`indentity`, `overlap`).

All `assembly` config parameter are given below with default values:

```
assembly:
  hybrid: true
  assembler: megahit
  mink: 25
  maxk: 99
  step: 4
  cap3:
    identity: 98
    overlap: 100
```

2.2.7 Analysis: gene annotation

In the IMP3 *Analysis* step, the predicted genes will be annotated using hmmer and a set of HMM databases. The **HMM databases** must be present in the *databases directory*. The database name given in the `hmm_DBs` field must match the directory containing the HMMs.

For the IMP3 *Binning* step using *Binny*, contigs needs to have been annotated with the essential genes HMMs (`essential`).

Different HMM databases have different names and *pre-calibrated cut-offs*. If the HMM database has cut-off values from hmmer calibration, set `cutoff` to `--cut_tc`. Otherwise, keep an empty string.

Regarding the naming convention, some HMM databases contain multiple models for the same functional entity (e.g. multiple HMMs for one KEGG KO). If the database contains models that are named with the functional entity + _ + another identifier, e.g. KO00033_1, the user can set **trim** to `--trimall` and IMP3 will remove the underscore and last part of the identifier.

```
hmm_DBs: "KEGG essential Pfam_A Resfams Cas dbCAN metacyc SwissProt TIGRPFAM"
hmm_settings:
  KEGG:
    cutoff: ""
    trim: "--trimall"
  essential:
    cutoff: "--cut_tc"
    trim: ""
  metacyc:
    cutoff: ""
    trim: "--trimall"
  Cas:
    cutoff: ""
    trim: ""
  Pfam_A:
    cutoff: "--cut_tc"
```

(continues on next page)

(continued from previous page)

```
    trim: ""
  dbCAN:
    cutoff: ""
    trim: ""
  Resfams:
    cutoff: ""
    trim: ""
```

2.2.8 Analysis: proteomics analysis preparation

In the IMP3 *Analysis* step, IMP3 provides several files for downstream **metaP** analysis.

- The user can use the protein sequence output of prokka directly as a search database for metaP analysis (not recommended).
- IMP3 also provides a file with Metaproteomics Analyzer-conformant FASTA file headers, filtered to proteins having a minimum user-defined number of tryptic peptides (`filter_N_peptides`), with incomplete non-tryptic ends cut away.
- IMP3 will add a FASTA file with host (or other reference) protein sequences (which should be placed in the *databases directory*). The provided host protein FASTA file should be consistent with the requirements of the proteomics software the user intends to use.
- IMP3 will optionally (`insert_variants: true/false`) generate a FASTA protein sequence file with the protein isoforms by inserting variants called on the assembled contigs (but *beware*).

```
proteomics:
  filter_N_peptides: 2
  host_proteome: "hostproteomefile"
  insert_variants: false
```

2.2.9 Analysis: mapped reads per gene / function

In the IMP3 *Analysis* step, featureCounts is used count the numbers of **metaG** and/or **metaT** reads mapping to coding sequences and other annotated functional categories that can be used for downstream differential expression / abundance analyses. Per default, IMP3 assumes **metaG** reads to **not be strand-specific** `mg: 0` and **metaT** reads to be **stranded in the truSeq way** `mt: 2`.

```
featureCountsStranding:
  mt: 2
  mg: 0
```

2.2.10 Taxonomy

In the IMP3 **:ref:'Taxonony <step_taxonomy> step, IMP3 runs 'kraken2 <https://ccb.jhu.edu/software/kraken2/>'_** and mOTUs2 on the **metaG** reads. While the mOTUs2 databases are installed with the *IMP3 installation*, the user needs to have a kraken2 database available in the *databases directory* and give its name under `krakendb`.

If the *Taxonony* and *Analysis* steps are performed, IMP3 will search the predicted genes by default for the mOTUs2-marker genes. The default COGs can be customized (field `COGS`).

```
krakendb: minikraken2
COGS: "COG0012 COG0018 COG0215 COG0525 COG0541 COG0016 COG0172 COG0495 COG0533 COG0552
↪"
```

2.2.11 Binning

In the IMP3 *Binning* step, the user can choose to run any combination of three different `binners`: MaxBin2, MetaBAT2, *Binny*. The results will then refined using DAStool.

For MaxBin2 and *Binny*, **metaG** reads are required. If there are only **metaT** reads, only MetaBAT2 will run.

Each binner comes with a few parameters that the user can configure:

- The minimum contig lengths `cutoff` for MetaBAT2 is **1500**.
- The default minimum contig lengths `cutoff` for MaxBin2 is **1000**.
- *Binny* uses the minimum contig lengths cutoff of `vizbin` (default **1000**).
- If VizBin has to be chosen.

```
binning:
  binners: "MaxBin MetaBAT binny"
  MaxBin:
    cutoff: 1000
  MetaBAT:
    cutoff: 1500
  binny:
    pk: 10
    nn: 4
    cutoff: 1000
  vizbin:
    dimension: 50
    kmer: 5
    size: 4
    theta: 0.5
    perp: 30
    cutoff: 1000
```

2.3 IMP3 input options

IMP3 is designed to perform integrated analyses of **metaG** and **metaT** data. Inputs can be either raw sequencing reads or already assembled contigs and mapped reads.

A typical *workflow* starts with a pair of files of paired-end **metaG** reads and a pair of **metaT** reads (both in FASTQ format, either gzipped or not). Alternatively, IMP3 can take only **metaG** or only **metaT** reads.

If the data is already assembled and the contigs should be annotated and binned into metagenomics-assembled genomes (**MAGs**), IMP3 also takes assemblies (in FASTA format) in addition to alignments (BAM files) or trimmed reads (FASTQ format) as input.

All inputs are defined in the *config file*.

2.3.1 Reads

The `Metagenomics` and `Metatranscriptomics` input fields expect two or three FASTQ or gzipped FASTQ files, separated by a space. The first two files should be the forward and reverse reads, if IMP3 analysis should start from pre-processing reads. The reads need to be in the same order in both files.

For processing the original raw Fastq files, the following setting should be used:

```
raws:
  Metagenomics: "/path/to/metagenomics.read.r1.fastq.gz /path/to/metagenomics.read.r2.
↪fastq.gz"
  Metatranscriptomics: "/path/to/metatranscriptomics.read.r1.fastq.gz /path/to/
↪metatranscriptomics.read.r2.fastq.gz"
  LongReads: ""
  LongReadTech: ""
  Contigs: ""
  Alignment_metagenomics: ""
  Alignment_metatranscriptomics: ""
```

If the user has **already pre-processed** reads, either for the *Assembly* step or for further analysis, the user is required to pass THREE files, two paired read files and one additional single ends file. The singleton read file is given last.

Note: *In this case, the user should NOT include* `preprocessing` *in the IMP steps*.

```
raws:
  Metagenomics: "/path/to/processed_metagenomics.read.r1.fastq.gz /path/to/processed_
↪metagenomics.read.r2.fastq.gz /path/to/processed_metagenomics.read.se.fastq.gz"
  Metatranscriptomics: "/path/to/processed_metatranscriptomics.read.r1.fastq.gz /path/
↪to/processed_metatranscriptomics.read.r2.fastq.gz /path/to/processed_
↪metatranscriptomics.read.se.fastq.gz"
  LongReads: ""
  LongReadTech: ""
  Contigs: ""
  Alignment_metagenomics: ""
  Alignment_metatranscriptomics: ""
```

The user can also give only **metaG** or only **metaT** reads to IMP3, either raw or pre-processed.

If the user wishes to perform a **metaG** assembly including long reads, a single file of **already pre-processed** long-reads data (in Fastq format). In this case, additionally the sequencing method (possible values are `nanopore` and `pacbio`) can be added.

Note: *The long reads are only used by IMP3, if* `metaspades` *is chosen as assembler.*

```
raws:
  Metagenomics: "/path/to/metagenomics.read.r1.fastq.gz /path/to/metagenomics.read.r2.
↪fastq.gz"
  Metatranscriptomics: ""
  LongReads: "/path/to/longread/reads.fastq"
  LongReadTech: ""
  Contigs: ""
  Alignment_metagenomics: ""
  Alignment_metatranscriptomics: ""
```

2.3.2 Contigs and alignments

If the user has already an assembly and would like to use IMP3 to annotate genes, perform binning and/or determine contig-level taxonomy, the contigs can be used as input in FASTA format (NOTE: the FASTA header should only

contain the contig name, NO other information and NO spaces: >CONTIGNAME1). In addition, the user can give reads either in FASTQ files or already aligned as BAM files. For FASTQ files, the same limitations apply as discussed above. The BAM files should be sorted by contig name and coordinate, and again, the contig names are NOT allowed to contain spaces !!!).

```
raws:
  Metagenomics: ""
  Metatranscriptomics: ""
  LongReads: ""
  LongReadTech: ""
  Contigs: "/path/to/assembly.fasta"
  Alignment_metagenomics: "/path/to/metagenomics/read.sorted.alignment.bam"
  Alignment_metatranscriptomics: "/path/to/metatranscriptomics/read.sorted.alignment.
↪bam"
```

2.4 Cluster configuration

Snakemake workflows, and hence IMP3, can be run in cluster mode, i.e. Snakemake will submit all the rules (except if explicitly stated) to a batch submission system. For Snakemake to know how to submit to the cluster, the command and arguments for submission can be stated in the `snakemake` call (here for a slurm batch scheduling system):

```
snakemake -s /path/to/IMP3/Snakefile --configfile /path/to/sample.config.yaml --use-
↪conda --conda-prefix /path/to/IMP3/conda --cluster-config /path/to/IMP3/config/
↪slurm.config.yaml --cluster "sbatch -t{params.runtime} --mem-per-cpu {params.mem} -
↪n {threads} -p batch"
```

The parameters (given in **brackets {}**) are predefined and will be taken from the `params` section of each Snakemake rule, so the user do not need to set them for every workflow task. More generally, a **cluster configuration file** (e.g. `slurm.config.yaml`) can be used. Currently, IMP3 currently supplies cluster config files for slurm , for PBS and for the (Oracle) grid engine.

If other scheduling systems are used, the **cluster configuration file** has to be adjusted. For help, please contact IMP support.

```
snakemake -s /path/to/IMP3/Snakefile --configfile /path/to/sample.config.yaml --use-
↪conda --conda-prefix /path/to/IMP3/conda --cluster-config /path/to/IMP3/config/
↪slurm.config.yaml --cluster "{cluster.call} {cluster.runtime}{params.runtime}
↪{cluster.mem_per_cpu}{params.mem} {cluster.threads}{threads} {cluster.partition}
↪{cluster.nodes}"
```

A short description for the IMP3 **cluster configuration file** is given below:

```
__default__:
      call: "sbatch"
      nodes: ""
      mem_per_cpu: "--mem-per-cpu "
      partition: "-p batch"
      quality: "-qos qos-batch"
      runtime: "-t"
      threads: "-n"

mg_filtering:
      nodes: "-N 1"
      partition: "-p bigmem"
      quality: "-qos qos-bigmem"
```

The **cluster configuration file** is in yaml format. There is a section about the default submit command at the top. In addition, the steps that require more RAM (`bigmem`) are listed, together with the arguments to achieve `bigmem` submissions. The user needs to fill in the command for submission and the way arguments are stated - we suggest using one of the provided cluster config files (`config/slurm.config.yaml`, `config/slurm_simple.config.yaml`, `config/pbs.config.yaml`, or `config/sge.config.yaml`) as template. Please consider submitting a pull request on the IMP3 gitlab repository for new cluster config files.

2.5 External databases

IMP3 makes use of external databases. They should be provided in or linked to one directory. See the install.

The following steps access external databases:

- Preprocessing:
  - adapter file for trimmomatic,
  - one or more genomes to filter against (optional),
  - sortmeRNA Silva database (metatranscriptomics only).
- Taxonomy:
  - a kraken2 database,
  - the GTDB-tk database (only, if binning is done),
  - the eukdetect (BUSCO) database (optional).
- Annotation:
  - HMM collections.

CHAPTER 3

Overview of IMP3 steps

3.1 IMP3 steps: Preprocessing

Preprocessing of reads consists of 1 to 3 steps: trimming, removal of ribosomal RNAs, filtering of reads mapping to one or more reference genomes.

3.1.1 Trimming

Trimming is performed on both metaG and metaT reads. Trimming is performed by Trimmomatic. Trimmomatic trimming includes the removal of the defined adapters, removal of low-quality bases at the beginning and/or ends of the reads, and/or truncation of reads if the quality in a sliding window becomes too low, and/or complete removal of a read if the remaining length is too short. Trimmomatic may produce singletons from paired-end data, when one read is completely removed due to quality reasons. The singletons produced from the first and second reads are concatenated into one file. After the Trimmomatic step, there are therefore always three output files for paired-end data, `r1`, `r2`, and `se`.

A user-defined step is the removal of trailing Gs that are commonly introduced by Nextseq machines when the sequenced DNA fragment is shorter than the number of bases added during the sequencing run. This is accomplished by cutadapt with setting `--nextseq-trim` and is performed on all reads, if requested in the config file.

3.1.2 rRNA removal

rRNA reads are separated from other metaT reads by SortMeRNA. The reason is that rRNA is highly abundant in total rRNA but doesn't assemble readily with the default settings of the assemblers in IMP3. Very commonly, rRNA is actually depleted during library preparation, with different success for different source organisms, making the rRNA abundance even less interpretable. While the rRNA removal step removes the rRNA reads from futher processing, they are kept in separates file for potential use outside of IMP3.

3.1.3 Reference filtering

Commonly, users will want to remove reads that map to one or more reference genomes, e.g. a host genome in a gut microbiome or a known contaminant. IMP3 achieves this step for metaG and metaT reads by mapping against the files chosen by the user with BWA. Only reads that do not map are kept. BWA is actually run independently on the paired-end data and singletons. Both partners of a set of paired reads where one partner maps to the host genome are removed from the final data set. If the user supplies more than one reference genome for filtering, the reads that did not map to the first reference will be mapped against the second. The reads that did not map to this one will be mapped against the third reference and so on. Currently reads that map to the reference genome(s) are not kept, nor are the alignments.

3.1.4 Cleaning up

Input and intermediary FASTQ files are gzipped at the end of the preprocessing step.

3.2 IMP3 steps: Assembly

IMP3 uses the iterative assembly approach of the original IMP. Like IMP, IMP3 can perform hybrid assemblies of metagenomic and metatranscriptomic reads. In addition, IMP3 can perform purely metagenomic reads, ignoring metatranscriptomics reads. IMP3 can also perform another kind of hybrid assembly, namely of short and long metagenomic reads. Obviously, IMP3 also performs iterative assemblies of only metagenomic or only metatranscriptomic reads.

After the assembly, reads are mapped back to the assembly.

3.2.1 Iterative assembly: metagenomic OR metatranscriptomic reads

In the simplest case, the user provides only one kind of short reads (metaG or metaT). IMP will then use the assembler defined by the user (Megahit or MetaSpades) to try to assemble all reads. After the assembly, the reads are mapped back to the assembled contigs. Reads that did not map will be given to the same assembler again. The set of contigs from the second assembly will be merged with the first set using the overlap assembler CAP3. The IMP developers have shown that this approach leads to longer contigs without compromising assembly quality. Finally, the metaG or metaT are mapped back to the final set of contigs by BWA.

The same approach is performed for the metaG reads, if the user provides metaG and metaT but does not choose `hybrid` as assembly option. In this case, both metaG and metaT reads are mapped to the final metagenomic assembly.

3.2.2 Hybrid assembly: long and short metagenomic reads

One of the assemblers built into IMP3, MetaSpades, is able to perform long/short-hybrid assemblies. If the user provides both kinds of metagenomic reads, the two sets of reads are co-assembled in the first round of the iterative assembly. Only the short reads are mapped back to the assembled contigs and, as described *above*, a second assembly will be attempted with the unmapped reads, and both sets of contigs will eventually be merged.

3.2.3 Iterative multi-omic hybrid assembly: metagenomic and metatranscriptomic reads

This approach developed for the original IMP. It is currently only implemented for the Megahit assembler. First, the metatranscriptomic reads are assembled. The metatranscriptomic reads are mapped back to the contigs using BWA. Reads that don't map are extracted and assembly is attempted a second time. The contigs from both assemblies are

supplied as reads to the hybrid assembly, together with all metagenomic and metatranscriptomic reads. All metagenomic and metatranscriptomic reads are mapped against the resulting contigs. Metagenomic and metatranscriptomic reads that don't map are extracted and are co-assembled. The resulting contigs are merged with the first set of hybrid contigs using the CAP3 overlap assembler.

Finally, both metaG and metaT reads are mapped back to the final set of contigs.

3.3 IMP3 steps: Analysis

In the Analysis step, IMP3 calls open reading frames, rRNA and tRNA genes, and annotate CRISPR repeats. Open reading frames are functionally annotated. The number of reads mapping to each gene and functional group of genes are calculated. The steps are described in more detail below.

3.3.1 Customized prokka

IMP3 uses prokka to call open reading frames (ORF), rRNA and tRNA genes, and annotate CRISPR repeats. Internally, prokka calls prodigal for the ORF calling, barrnap for the rRNA regions, ARAGORN for tRNA loci and MinCED to detect CRISPR arrays.

Prokka forces prodigal to only call complete genes. Due to the fragmented nature of metagenomic contigs, it is preferable to also allow partial genes. IMP3's customized prokka allows prodigal to call incomplete ORFs and records whether prodigal detected start and stop codons. One side-effect of this is that the amino acid sequences prokka returns `prokka.faa` are badly formatted. The prokka amino acid sequences also don't start with M if prodigal called genes with an alternative start codon. Both issues are corrected in another IMP3 step as part of the metaproteomics preparations (`proteomics.proteins.faa`).

Prokka would usually provide some functional analyses by aligning the called ORFs to some databases. However, this analysis is optimized for speed, meaning that genes that have been annotated with one database are not annotated with the next. This leads to genes potentially having inconsistent annotations, and it would be impossible for the user to find out what would have been the best hit in another database. Since IMP3 does *functional annotations* of all genes with any database the user chooses to reach consistent annotations, we've disabled the prokka-based annotation.

Prokka also spends considerable time to convert its output into genbank format. As IMP3 has no need for genbank-formatted data, we've disabled this.

3.3.2 Annotation with HMMs

3.3.3 Variants for metaP searches

3.4 IMP3 steps: Binning

3.4.1 Binny

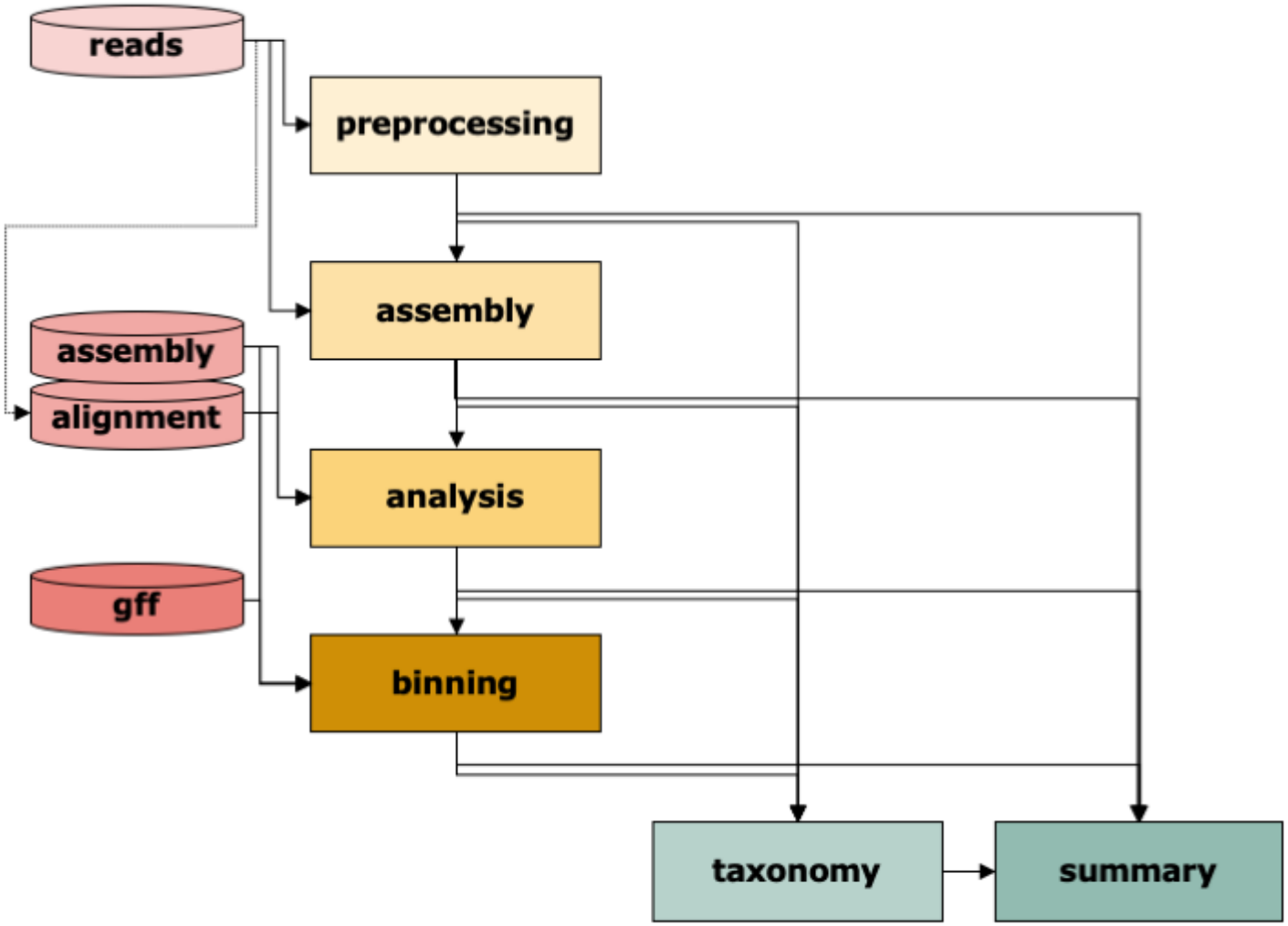
3.5 IMP3 steps: Taxonomy

Additionally, kraken2 is run on the contigs from the *Assembly* step.

3.6 IMP3 steps: Summary

IMP3 will summarize…

The IMP3 workflow consists of six modules: *Preprocessing* of reads , *Assembly*, *Analysis* of contigs including functional annotation, *Binning* of contigs, determination of *Taxonomy* at various levels, and *Summary* including visualization. Typically, IMP3 will use raw sequencing reads as *input* and run through the whole workflow.



Given the appropriate *input*, most steps can be run independently. For instance, if the user has already processed reads and wants to assemble and annotate them. The user would then select only `assembly` and `analysis` from the *steps* (and potentially `summary`). The user can also supply directly a given assembly to annotate and bin it. Or an already annotated assembly just for binning etc. Just enter the appropriate *input* data and define the `steps` in the *config* file and IMP3 will take care of it.

To get an impression of all Snakemake steps of a run with **metaG** and **metaT** reads as input using **hybrid assembly**, download `this file`.

CHAPTER 4

Overview of IMP3 output

4.1 The IMP3 report

4.2 IMP3 stats, summaries and visualizations

And what's all the other clutter in the Stats directory? Find more details on the stats recorded during the run in the sections on

***assembly outputs*, *analysis outputs*,** *binning outputs*, and *taxonomy outputs*.

4.3 IMP3 output: Preprocessing

The *Preprocessing step* creates output files in the `Preprocessing` and `Stats` directories within the defined `outputdir`. The exact naming of the files depends on the *configuration* settings.

For all runs with reads as *input*, the original input reads will be copied to the `Preprocessing` directory and renamed into `<mg|mt>.<r1|r2>.fq.gz`. **The final set of processed reads** (that will be the input for the *Assembly step*) will be pointed to by symbolic links `<mg|mt>.<r1|r2|se>.preprocessed.fq`*[0] ,†[0]. All IMP3 runs with the step *Preprocessing* will also have the trimmed reads processed with trimmomatic (`<mg|mt>.<r1|r2|se>.trimmed.fq.gz`).

MetaT reads are always filtered to remove rRNA reads using SortMeRNA . The remaining reads from this filtering step are named `mt.<r1|r2|se>.trimmed.rna.fq.gz`. If reads were mapped against one or more reference genomes, the filtered reads are named `mt.<r1|r2|se>.trimmed.rna_filtered.<reference>_filtered.fq.gz` and/or `mg.<r1|r2|se>.trimmed.<reference>_filtered.fq.gz`.

Note: *Why is there a Preprocessing directory if you did not choose to do preprocessing?* If already preprocessed reads were given as *input*, these reads will be placed in the `Preprocessing` directory following the exact same

[0] `mg` denotes **metaG** data throughout the workflow, `mt` represents **metaT** data.

[0] `r1` contains first reads with a partner, which will be in `r2`, `se` contains the single ends that have lost their partner during the *Preprocessing step*.

naming conventions as described above, to ensure that the downstream steps can use the same consistent input files and directories.

4.3.1 Stats from Preprocessing step

The *Preprocessing step* will also collect some statistics on the original and processed reads and save them to the **Stats** directory. To this end, it runs FastQC on the original and preprocessed reads. The resulting HTML reports and directories with more detailed tables are in the respective subdirectories `<mg|mt>/` named `<mg|mt>.<r1|r2>_fastqc.<zip|html>` and `<mg|mt>.<r1|r2|se>.preprocessed_fastqc.<zip|html>`. In addition, the number of reads in each step is counted in `<mg|mt>/<mg|mt>.read_counts.txt`, which holds two tab-separated columns with the file names and number of reads, respectively.

4.4 IMP3 output: Assembly

During the *Assembly step* (or if the user provided an existing assembly, see *input*) all outputs will be written to the `Assembly` directory within the defined `outputdir` directory (see *configuration*).

- The final output is a FASTA file of the assembled contigs `<mg|mt|mgmt>.assembly.merged.fa` [*] . The FASTA headers contain the **sample name** as given in the *config file*, separated by an underscore, *contig*, another underscore, and a number, e.g. *test_contig_1*.
- The **index** files of the final contig file will be generated , namely for BWA (suffixes `amb`, `ann`, `bwt`, `pac`, and `sa`), Samtools (`fai`) and bioperl (`index`).
- A `bed3` file is also stored for later access.

Note: *some of these files are produced by the Analysis step , so they will not be present after running only the Assembly step*.

During the *Assembly step*, IMP3 maps back the **processed** reads to the final contigs and stores the alignment `<mg|mt>.reads.sorted.bam` and index `<mg|mt>.reads.sorted.bam.bai`. The BAM files are sorted by contig name and position.

The *Assembly step* actually consists of a large number of sub-steps (*iterative assembly*) generating a huge amount of intermediate result files and directories that will be archived and compressed into `intermediary.tar.gz`.

4.4.1 Stats from Assembly step

The *Assembly step* will also collect some summary statistics and save them to the `Stats` directory:

- The GC-content of the final contigs is recorded in the tab-separated file `<mg|mt|mgmt>/<mg|mt|mgmt>.assembly.gc_content.txt`, which contains a header and holds two columns for the contig names and GC in percent (i.e 0-100), respectively.
- The length of the contigs is provided in a tab-separated file `<mg|mt|mgmt>/<mg|mt|mgmt>.assembly.length.txt` with two simple columns, contig names and lengths.
- The stats on the read mapping are kept in the `Stats` subdirectories with the **metaG** and/or **metaT** (`mg/` or `mt/`):
- `<mg|mt>/<mg|mt|mgmt>.assembly.contig_flagstat.txt` contains the numeric part of the samtools flagstat output.
- the average depth of coverage for each set of reads for all contigs that have at least one read mapping to them is given in `<mg|mt>/<mg|mt|mgmt>.assembly.contig_depth.txt`. This file is headerless and tab-separated, with the contig names in the first column and the average depth of coverage in the second.

- The *Binning step* adds based on this the file `<mg|mt>/<mg|mt|mgmt>.assembly.contig_depth.0.txt`, which also contains lines for contigs with zero coverage.
- **The file `<mg|mt>/<mg|mt|mgmt>.assembly.contig_coverage.txt` contains the processed output of bedtools ge**
  - 1: contig names,
  - 2: 0 (= the first position in bed coordinates),
  - 3: contig length (= the last position in bed coordinates),
  - 4: the number of regions overlapping with the aligned positions in the bam file (roughly = number of mapping reads),
  - 5: the number of covered positions,
  - 6: the number of positions (= value in column 3),
  - 7: the coverage breadth, i.e. proportion of covered positions (scaling 0-1)).

4.5 IMP3 output: Analysis

4.5.1 Annotation

IMP3 writes the results of the *Analysis* into the `Analysis` directory within in the defined `outputdir`, see *configuration*.

The gene annotation results are in the `annotation` subdirectory.

The GFF file

The GFF3 file `annotation_CDS_RNA_hmms.gff` is the final annotation file and contains all gene annotations (including mRNAs, rRNAs, tRNAs and CRISPR arrays) with the functional annotations and gene descriptions. The file is in standard GFF format with nine columns:

- 1: contig ID,
- 2: source software,
- 3: kind of feature ("CDS" for protein-coding regions, "rRNA" for rRNAs, "tRNA" for tRNAs, and "tmRNA" for tRNAs with coding regions),
- 4: left-most coordinate of feature (1-based),
- 5: right-most coordinate of the feature (1-based, inclusive),
- 6: a score ("." if no score is reported),
- 7: the feature's direction or sense ("+" or "-"),
- 8: frame (0 for CDS, "." for other features),
- 9: attributes (each attribute starts with a key followed by "=", e.g. "ID=", attributes are separated by ";")

The most important attributes of column *9* are:

- **ID**, which is used in other gene-based outputs,
- **partial, which contains information from the gene-caller (Prodigal) about whether the open reading frames are complete:**

  - *00* means both start and stop codons were found,
  - *11* means neither were found,
  - *01* means the right-most end is incomplete - i.e. missing stop-codon for +strand features, missing start-codon for -strand features,
  - *10* means the left-most end is incomplete - i.e. missing start-codon for +strand features, missing stop-codon for -strand features, and

- **the results of the HMM searches**, named with the HMM database name provided in the **config file**, e.g. *essential*= for the essential genes.

The functional annotation of the genes by HMMer produces a number of intermediary outputs. With the results summarized in `annotation_CDS_RNA_hmms.gff`, the intermediary files are archived and compressed in `intermediary.tar.gz`. ..... Reads per gene / group .....

The annotated features are used to determine the numbers of reads mapping to the features and to groups of features that share the same functional annotation using featureCounts:

- The results are written to `<mg|mt>.<CDS|rRNA>_counts.<tsv>` for single features and to `<mg|mt>.<database>.counts.<tsv>` for the functional groups.
- **The format of these files is unchanged featureCounts output, i.e. there's a line starting with "#" that contains the original**
  - 1: feature or group ID,
  - 2: contig or contigs, separated by ";",
  - 3: left-most coordinate (as described for the .gff format) or coordinates (separated by ";"),
  - 4: right-most coordinate (as described for the .gff format) or coordinates (separated by ";"),
  - 5: feature direction or directions (separated by ";"),
  - 6: total length of feature(s), 7: total counted reads).
- In addition, summaries of the numbers of reads that were mapped and overlapped with the features are found in the respective files ending on `tsv.summary`.
- If only **metaT** reads were used as input, there will be no data for rRNA, because rRNA has been filtered out.

Proteomics databases and gene sequences

IMP3 output can be used for downstream **metaP** analysis:

- `proteomics.final.faa`: the FASTA file for downstream **metaP** analysis. The FASTA header for the protein sequences is in a format that

should work with most proteomics search engines and analysis tools, in particular the MetaProteomeAnalyzer . - As an intermediary step (without host proteins), a file named `proteomics.proteins.faa` is generated.

The proteomics file is a cleaned version of `prokka.faa`. prokka outputs a few more files: - `prokka.ffn` with the CDS, - `prokka.fna` with the contigs (also present in `prokka.fsa` with a slightly different header), - `prokka.log`, a logfile, - `prokka.txt`, a summary of the number of analysed contigs and annotated features, - `prokka.tsv`, a tabular output with all features, - `prokka.tbl`, a sort-of flat version of the same information, - `prokka.gff`, a GFF file with lots of commented lines (starting with "#"), which is actually the foundation of the GFF file described above.

An intermediary step between this file and the final GFF is `annotation.filt.gff` which contains all the information of the original prokka output minus the comments. Depending on the planned further analysis steps, the user may also see indices for some of the sequences (`prokka.<faa|ffn>.<suffix>`). If the IMP3 *Binning* is run,

IMP3 will add a file containing the link between the gene IDs as given by prokka and the format required by DASTool, called `annotation_CDS_RNA_hmms.contig2ID.tsv`.

4.5.2 SNPs

IMP3 uses samtools mpileup to call variants. Samtools mpileup records positions where mapping reads differ from the assembly consensus. The information is stored in the `snps` subfolder of the `Analysis` directory in VCF and tabix formats: `<mg|mt>.variants.samtools.vcf.gz` with the index file `<mg|mt>.variants.samtools.vcf.gz.tbi`.

4.5.3 Taxonomy

The `Analysis` directory contains the output of the *taxonomy* step. The exception is the output of GTDBtk, which is run on the bins selected by DASTool. See the *next* section for details.

4.5.4 Stats from the Analysis step

In addition to the annotation and SNP calling, the analysis step will also collect some stats for you and save them to the **Stats** directory.

4.6 IMP3 output: Taxonomy

The output of the *taxonomy step* is mostly written to `taxonomy` in the **Analysis** directory. The exception is the output of GTDBtk, which is run on the bins selected by DASTool and has its results summarized with the bins.

4.6.1 Stats from the taxonomy step

The *Taxonomy step* does not always directly write to the `Stats` directory. Some of the results are summarized by the *Summary*. The GTDBtk results are summarized together with the results from the *binning step* results in `Stats/<mg|mt|mgmt>/<mg|mt|mgmt>.bins.tsv`.

4.7 IMP3 output: Binning

The IMP3 *Binning step* will output the results of all binners the user selected, the results from DASTool and the results of running GRiD on the bins that DASTool selected in the `Binning` directory within the `outputdir` (see *configuration*).

4.7.1 Binning tools

Each binninng tool has a separate output directory within the `Binning` directory. Currently, IMP3 implements MetaBAT2, MaxBin2 and *Binny*. Each binning tools generates a final file linking contigs to bins that is always named `scaffold2bin.tsv`.

`MetaBAT` is the output directory of MetaBAT2. MetaBAT2 outputs a tab-separated file with contig IDs and bin numbers `metabat_res`, which is linked to in `scaffold2bin.tsv`. In addition, the contigs in each bin, except bin *0*, are given in one *Fasta* file `metabat_res.<bin>.fa`.

`MaxBin` is the output directory of MaxBin2. *MaxBin2 <https://kbase.us/applist/apps/kb_maxbin/run_maxbin2/release>'_outputs a tab-separated file with contig IDs and bin names ''maxbin_contig2bin.txt'*, which is linked to in `scaffold2bin.tsv`. MaxBin2 also outputs a *Fasta* file for each bin `maxbin_res.<bin>.fasta`. Contigs which can't be put in a bin are in `maxbin_res.noclass` and contigs that are too short for analysis are in `maxbin_res.tooshort`. Summaries of the essential markers MaxBin detected in each bin and the bin size are in `maxbin_res.marker` and `maxbin_res.summary`. In addition, intermediary results are in `maxbin_res.marker_of_each_bin.tar.gz` and a log is recorded to `maxbin_res.log`.

`binny` is the output directory of *Binny*. *Binny* outputs its final result `contigs2clusters.10.4.tsv` and `contigs2clusters.10.4.RDS`. The bins with at least some completeness (names starting with P ("perfect"), G ("good"), O ("okay"), L "low completeness") are extracted and copied to `scaffold2bin.tsv`. *Binny* is based on tSNE embeddings by VizBin of *k*-mer frequencies in contigs with masked (temporarily removed) rRNA genes (`<mg|mt|mgmt>.assembly.merged.cut.fa`) and the coordinates from VizBin are stored in `<mg|mt|mgmt>.vizbin.with-contig-names.points`. For comprehensibility and transparency, intermediate results are kept in `clusterFiles.tar.gz`, `clusterFirstScan.<pk>.<nn>.tsv`, `bimodalClusterCutoffs.<pk>.<nn>.tsv`, `reachabilityDistanceEstimates.<pk>.<nn>.tsv`, `clusteringWS.<pk>.<nn>.Rdata`, and `binny_WS.Rdata`. *Binny* also visualizes its intermediary results in `scatterPlot<1-4>.<pk>.<nn>.pdf` and the final bins visualized in a tSNE embedding in `finalClusterMap.10.4.png`.

4.7.2 DASTool

The summarized output of DASTool is in `selected_DASTool_summary.txt` and the `selected_DASTool_bins` directory. The contigs of each bin are in `selected_DASTool_bins/<bin>.contigs.fa`. DASTool uses the presence of single-copy marker genes to assess bins. As IMP3 has already generated gene predictions in the *Analysis step*, they are used as input for DASTool (with a changed header; `prokka.renamed.faa`) and DASTool keeps the annotations per gene in `prokka.renamed.faa.<archaea|bacteria>.scg`. The lengths of all contigs are in `selected.seqlength`. The results of DASTool assessment are in `selected_<binner>.eval`. DASTool also gives visual output in `selected_DASTool_hqBins.pdf` and `selected_DASTool_scores.pdf` and a log file in `selected_DASTool.log`.

4.7.3 GRiD

The results of GRiD is given for every DASTool selected bin in `selected_DASTool_bins/<bin>/grid`. IMP3 archives and compresses these directories after adding the information into `Stats/<mg|mt|mgmt>/<mg|mt|mgmt>.bins.tsv`. All the archived data is in `per_bin_results.tar.gz`.

4.7.4 GTDBtk

If the *Taxonomy step* is selected in addition to the *Binning step*, the selected bins from DASTool are analysed with GTDBtk. The results are in `selected_DASTool_bins/<bin>/GTDB`. IMP3 archives this data in `per_bin_results.tar.gz`, after the results are added to `Stats/<mg|mt|mgmt>/<mg|mt|mgmt>.bins.tsv`.

4.7.5 Stats from the Binning step

The *Binning step* writes to the `Stats` directory. If no *Taxonomy step* is run, the GRiD results are summarized together with the rest of the binning results in `Stats/<mg|mt|mgmt>/<mg|mt|mgmt>.bins.tsv`. If taxonomy is performed, the GRiD results are also combined with the GTDBtk results in `Stats/<mg|mt|mgmt>/<mg|mt|mgmt>.bins.tsv`. Some more results are summarized by the Summary step if defined.

4.8 IMP3 logs and status files

4.8.1 Logs

4.8.2 Status files

4.8.3 The configuration file

4.8.4 The .snakemake

While running, IMP3 will write the intermediary output and final results into the user-defined **output directory** (`outputdir`, see *configuration*). Before finishing, IMP3 will compress some of the intermediary steps to reduce space. Finally, the IMP3 workflow will generate summaries and visualizations (see *configuration* if defined).

An overview of the all outputs, files and their directory structure is given below:

Preprocessing

Preprocessed reads:	<ome>.<read>.preprocessed.fq
Intermediary steps:	<ome>.<read>.<stage>.fq.gz
Original reads:	<ome>.<read>.fq.gz

Assembly

Assembly:	<omes>.assembly.merged.fa
- indices:	<omes>.assembly.merged.fa.<suffix>
Aligned reads:	<ome>. reads.sorted.bam
- indices:	<ome>. reads.sorted.bam.bai
Intermediary steps:	intermediary.tar.gz

Analysis

/taxonomy

Kraken2 results:	kraken/<ome>.kraken.<output|report>
Kraken2 contigs:	kraken/<ome>.kraken.contig.parsed.tsv
mOTU2 results:	mOTUs/<ome>. mOTU.counts.tsv
mOTU genes:	mOTU_links/bestPresentHitPhylogeny.tsv

/annotation

Final annotations:	annotation_CDS_RNA_hmms.gff
Feature counts funct.:	<ome>.<database>.counts.<tsv|tsv.summary>
Feature counts genes:	<ome>.<CDS|rRNA>_counts.<tsv|tsv.summary>
Proteomics DB:	proteomics.final.faa
- intermediary:	proteomics.proteins.faa
Prokka results:	annotation.filt.gff
- sequences:	prokka.<faa|ffn|fna|fsa>
- indices:	prokka.<faa|ffn|fna|fsa>.<suffix>
- others:	prokka.<suffix>
HMMer intermediary:	intermediary.tar.gz
DASTool-ID:	annotation_CDS_RNA_hmms. contig2ID.tsv

/snps

SNPs:	<ome>.variants.samtools.vcf.gz
- Indices:	<ome>.variants.samtools.vcf.gz.tbi

Binning

DASTool result:	selected_ DASTool_summary.txt
- contigs per bin:	selected_ DASTool_bins/<bin>.contigs.fa
- protein predictions:	prokka.renamed.faa
- single-copy ann.:	prokka.renamed.faa.<taxon>.scg
- binner inputs:	selected_<binner>.eval
- others:	selected_ DASTool_<data>.<suffix>
Binner outputs:	<binner>/
Binner final results:	<binner>/scaffold2bin.tsv
More results per bin:	per_bin_results.tar.gz

Stats

Table with all stats:	all_stats.tsv
R workspace of stats:	all_stats.Rdata
Table binning stats:	all_bin_stats.tsv
Read QC:	<ome>/<ome>.<read+stage>_fastqc.<suffix>
Read stats:	<ome>/<ome>.read_counts.txt
Assembly stats:	<omes>/<omes>.assembly.<stats>.txt
Mapping stats contigs:	<ome>/<omes>. assembly.contig_<stats>.txt
Mapping stats genes:	<ome>/annotation/<ome>.<stats>.<suffix>
Mapping stats binning:	<omes>/<omes>.bins.tsv

Visualization

Visualizations:	<step>_<plot>.png

Empty files that are touched when subprocesses have been finished

The user can run *reporting* to get more information on the run, together with the visualizations in HTML format. All outputs are explained in more detail in the next sections.

CHAPTER 5

FAQ

The FAQ is empty. Send a question via the git-lab .