metage2metabo Documentation

AuReMe

Apr 05, 2024

CONTENTS

Metage2metabo is a Python3 tool to perform graph-based metabolic analysis starting from annotated genomes (**reference genomes or metagenome-assembled genomes**). It uses **Pathway Tools** in a automatic and parallel way to *reconstruct metabolic networks* for a large number of genomes. The obtained metabolic networks are then *analyzed individually and collectively* in order to get the *added value of metabolic cooperation in microbiota* over individual metabolism, and to *identify and screen interesting organisms* among all.

M2M can be used as a whole workflow (`m2m workflow`, `m2m metacom`) or steps can be performed individually (`m2m recon`, `m2m iscope`, `m2m cscope`, `m2m addedvalue`, `m2m mincom`, `m2m seeds`).

CHAPTER

ONE

GENERAL INFORMATION ABOUT THE MODELLING

M2M has two main dependencies for modelling metabolic networks: MeneTools and Miscoto. Accordingly metabolic models in M2M follow the producibility in metabolic networks as defined by the network expansion algorithm. Mainly, two rules are followed:

- a *recursive rule*: the products of a reactions are producible if **all** reactants of this reaction are themselves producible
- an *initiation rule*: producibility is initiated by the presence of nutrients, called *seeds*.

A metabolite that is producible from a set of nutrients is described as being "in the scope of the seeds". The computation is made using logic solvers (Answer Set Programming). The present modelling ignores the stoichiometry of reactions (2A + B –> C is considered equivalent to A + B –> C), and is therefore suited to non-curated or draft metabolic networks, as the ones built using M2M with the PathoLogic software of Pathway Tools handled by Mpwt. Many works have relied on network expansion to study organisms (here, here or there) and communities (here, here, or here). It has been compared, combined to steady-state modelling (Flux Balance Analysis).

CHAPTER

TWO

DOCUMENTATION OF M2M

2.1 Requirements and Installation

2.1.1 Requirements

m2m uses Pathway Tools for the reconstruction of draft metabolic networks from annotated genomes.

- **Pathway-Tools version 22.5 or higher (free for academic users)**
  - **Pathway-Tools requirements**
    - **Linux**: Gnome terminal and Libxm4

```
apt-get update && apt-get install gnome-terminal libxm4
```

    - **All OS: NCBI Blast and a ncbirc file in user's home directory**
      - Install with apt-get

```
apt-get update && apt-get install gnome-terminal libxm4 ncbi-blast+
echo "[ncbi]\nData=/usr/bin/data" > ~/.ncbirc
```

      - Install with a dmg installer on MacOS
  - **Pathway-Tools install**
    - **Linux**

```
chmod +x ./pathway-tools-22.5-linux-64-tier1-install
./pathway-tools-22.5-linux-64-tier1-install
```

      and follow the instructions during the interactive install

      For a silent install:

```
./pathway-tools-22.5-linux-64-tier1-install --InstallDir your/install/
↪directory/pathway-tools --PTOOLS_LOCAL_PATH your/chosen/directory/for/
↪data/ptools --InstallDesktopShortcuts 0 --mode unattended
```

    - **MacOS**

      Install though a .dmg installer with a graphical interface.

    - **Warning**

/!\ For all OS, Pathway-Tools must be in `$PATH`.

On Linux and MacOS: `export PATH=$PATH:/your/install/directory/pathway-tools`.

Consider adding Pathway Tools in `$PATH` permanently by running

```
echo 'export PATH="$PATH:/your/install/directory/pathway-tools:"' >> ~/.
↪bashrc
```

Then source bashrc file:

```
source ~/.bashrc
```

m2m relies on several Python packages:

- mpwt to automatize metabolic network reconstruction with Pathway-Tools.
- padmet to manage metabolic networks.
- menetools to analyze individual metabolic capabilities using logic programming.
- miscoto to analyze collective metabolic capabilities and select communities within microbiota using logic programming.

A list of Python dependencies is available in requirements.txt

Developed and tested on Linux (Ubuntu, Fedora, Debian) and MacOs (version 10.14) with Python3.8.

Continuous Integration using GitHub Actions with Python3.8 and Python3.9 on ubuntu-latest, macos-latest and windows-latest (corresponding virtual environment).

2.1.2 Installation with pip

```
pip install Metage2Metabo
```

2.1.3 Installation with Docker

To create the m2m image, use the Dockerfile found in Recipes of the Github repository. Note that the **Pathway-Tools installer** and the **Oog.jar file** need to be placed in the same folder than the Dockerfile. The name of the installer file is currently hardcoded in the Dockerfile. Hence it must be changed if you use a different version of Pathway-Tools. Please note that the following commands (especially due to the use of root privileges) apply to Linux OS.

```
# Launch docker.
sudo systemctl start docker

" Build image locally.
sudo docker build -t my_image .
```

To create and launch the container in interactive mode:

```
sudo docker run -ti -v /my/path/to/my/data:/shared --name="my_container" my_image bash
```

Then you can exit the contanier with `exit`. You can launch again the container with:

```
sudo docker start my_container

sudo docker exec -ti my_container bash
```

2.1.4 Installation with Singularity (e.g. on a cluster)

Singularity with Pathway Tools

Singularity [Ku2017] can be used to launch m2m on a cluster. Please refer to the recipe of the Github repository of the project. The Singularity container has to be created from the recipe. You might need to do it on a personal computer since it requires administrator priviledges. To use the container on a cluster, the path to Pathway Tools ptools folder should be indicated in the recipe. Therefore, you have to replace `/external/folder/ptools` with the path where you want to put the ptools-local folder (which will contain the PGDB created by Pathway-Tools).

Like for the Dockerfile, Pathway-Tools installer is hardcoded in the recipe so if you use another version, you have to modify the recipe. And the **Pathway Tools installer** and **Oog.jar file** must be stored in the same folder than the Singularity recipe.

To create a container named m2m.sif:

```
sudo singularity build m2m.sif Singularity
```

If you have issue with disk space in the temporary folder used by Singularity, it is possible to specify another temporary folder with `SINGULARITY_TMPDIR`, such as:

```
sudo SINGULARITY_TMPDIR=/another/tmp/folder singularity build m2m.sif Singularity
```

Warning, this command will fail if folder `another/tmp/folder` does not exist.

To use Pathway-Tools, a .ncbirc file is required in the home directory, containing the path to Blast:

```
.ncbirc:

[ncbi]\nData=/usr/bin/data
```

Dealing with Pathway Tools ptools local folder. You might need an external ptools-local folder when working on a cluster. A solution is to create the ptools-local in a local folder then move it inside the Singularity container. Eventually, you have to move it outside the Singularity container after it has been built.

First, enter the Singularity container and mount the external folder:

```
singularity run -B /external/folder:/external/folder m2m.sif
```

Then move the ptools-local folder from the Singularity folder to the folder in your local environment.

```
cp -r /opt/ptools-local /external/folder
```

This will move the ptools-local folder (with permissions) from Singularity container to the local machine.

In this way, PGDBs can be stored in the folder outside your container.

Finally, you can launch jobs with the Singularity container by giving a sh file containg m2m commands.

```
m2m.sh:

m2m workflow -g genomes_dir -s seeds.sbml -o output_dir -c cpu_number
```

So you can encapsulate it in a sh script:

```
my_script.sh:
```

(continues on next page)

(continued from previous page)

```
#!/bin/bash

# Don't forget to source the Singularity environment if needed.
. /local/env/envsingularity.sh

singularity exec m2m.sif bash m2m.sh
```

This file can now be launched on a cluster, for example with SLURM [Yo2003]:

```
sbatch --cpus-per-task=4 --mem=8G my_script.sh
```

With some versions of Singularity (superior to 3.6) running `singularity exec m2m.sif bash /cluster/myspace/m2m.sh` will show the following error message:

```
/bin/bash: /cluster/myspace/m2m.sh: No such file or directory
```

This error comes from modifications in Singularity linked to security issue. Especially the paths accessible to a container have been reduced. To fix this the `-B` option can be used to give access to Singularity to a specific path, for example:

```
singularity exec -B /cluster/myspace/m2m:/cluster/myspace/m2m m2m.sif bash /cluster/
↪myspace/m2m.sh
```

The first path after `-B` option corresponds to the local/cluster path and the second path corresponds to the path inside the Singularity container.

Singularity without Pathway Tools

A Singularity without Pathway Tools container is available publicly at Singularity-Hub. As there is no Pathway Tools in this container, you can not use `m2m recon` and `m2m workflow` commands.

You can download the container with the command:

```
singularity pull shub://ArnaudBelcour/metage2metabo-metacom_singularity
```

With this container, you can call m2m commands like `m2m metacom`:

```
singularity exec metage2metabo-metacom_singularity_latest.sif m2m metacom ...
```

2.2 Commands

2.2.1 Outputs and logs

By default, `m2m` writes into the console (stdout) and into a log file located at the root of the results directory and named after the subcommand that was executed. The option `-q` can be given to any `m2m` subcommand to write in the log file and to stdout only the warnings, errors and critical issues, together with the path to the log file.

2.2.2 m2m Commands

Features

```
Copyright (C) Dyliss & Pleiade
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
m2m is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


usage: m2m [-h] [-v] {recon,iscope,cscope,addedvalue,mincom,seeds,workflow,
↪metacom,test} ...

From metabolic network reconstruction with annotated genomes to metabolic␣
↪capabilities screening to identify organisms of interest in a large␣
↪microbiota.
For specific help on each subcommand use: m2m {cmd} --help

options:
-h, --help            show this help message and exit
-v, --version         show program's version number and exit

subcommands:
valid subcommands:

{recon,iscope,cscope,addedvalue,mincom,seeds,workflow,metacom,test}
    recon               metabolic network reconstruction
    iscope              individual scope computation
    cscope              community scope computation
    addedvalue          added value of microbiota's metabolism over individual
↪'s
    mincom              minimal communtity selection
    seeds               creation of seeds SBML file
    workflow            whole workflow
    metacom             whole metabolism community analysis
    test                test on sample data from rumen experiments

Requires: Pathway Tools installed and in $PATH, and NCBI Blast
```

m2m recon

```
usage: m2m recon [-h] -g GENOMES -o OUPUT_DIR [-c CPU] [-q] [-l {2,3}] [--
↪noorphan] [-p] [--clean] [--pwt-xml]

Run metabolic network reconstruction for each annotated genome of the input␣
↪directory, using Pathway Tools

options:
-h, --help            show this help message and exit
-g GENOMES, --genomes GENOMES
                      annotated genomes directory
```

(continues on next page)

(continued from previous page)

```
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
-c CPU, --cpu CPU     cpu number for multiprocessing
-q, --quiet           quiet mode
-l {2,3}, --level {2,3}
                        Level for SBML creation, 2 or 3
--noorphan            use this option to ignore reactions without gene or␣
↪protein association
-p, --padmet          create padmet files
--clean               clean PGDBs if already present
--pwt-xml             use this option to use Pathway Tools xml (incompatible␣
↪with -p)
```

m2m iscope

```
usage: m2m iscope [-h] -n NETWORKS_DIR -s SEEDS -o OUPUT_DIR [-q] [-c CPU]

Compute individual scopes (reachable metabolites from seeds) for each␣
↪metabolic network of the input directory

options:
-h, --help            show this help message and exit
-n NETWORKS_DIR, --networksdir NETWORKS_DIR
                        metabolic networks directory
-s SEEDS, --seeds SEEDS
                        seeds (growth medium) for metabolic analysis
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
-q, --quiet           quiet mode
-c CPU, --cpu CPU     cpu number for multiprocessing
```

m2m cscope

```
usage: m2m cscope [-h] -n NETWORKS_DIR -s SEEDS -o OUPUT_DIR [-m MODELHOST] [-
↪q] [-t TARGETS]

Compute the community scope of all metabolic networks

options:
-h, --help            show this help message and exit
-n NETWORKS_DIR, --networksdir NETWORKS_DIR
                        metabolic networks directory
-s SEEDS, --seeds SEEDS
                        seeds (growth medium) for metabolic analysis
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
-m MODELHOST, --modelhost MODELHOST
                        host metabolic model for community analysis
-q, --quiet           quiet mode
```

(continues on next page)

(continued from previous page)

```
-t TARGETS, --targets TARGETS
                        Optional targets for metabolic analysis, if not used␣
↪metage2metabo will use the addedvalue of the community
```

m2m addedvalue

```
usage: m2m addedvalue [-h] -n NETWORKS_DIR -s SEEDS -o OUPUT_DIR [-m␣
↪MODELHOST] [-q]

Compute metabolites that are reachable by the community/microbiota and not by␣
↪individual organisms

options:
-h, --help            show this help message and exit
-n NETWORKS_DIR, --networksdir NETWORKS_DIR
                        metabolic networks directory
-s SEEDS, --seeds SEEDS
                        seeds (growth medium) for metabolic analysis
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
-m MODELHOST, --modelhost MODELHOST
                        host metabolic model for community analysis
-q, --quiet           quiet mode
```

m2m mincom

```
usage: m2m mincom [-h] -n NETWORKS_DIR -s SEEDS -o OUPUT_DIR [-m MODELHOST] [-
↪q] -t TARGETS

Select minimal-size community to make reachable a set of metabolites

options:
-h, --help            show this help message and exit
-n NETWORKS_DIR, --networksdir NETWORKS_DIR
                        metabolic networks directory
-s SEEDS, --seeds SEEDS
                        seeds (growth medium) for metabolic analysis
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
-m MODELHOST, --modelhost MODELHOST
                        host metabolic model for community analysis
-q, --quiet           quiet mode
-t TARGETS, --targets TARGETS
                        targets for metabolic analysis
```

m2m workflow

```
usage: m2m workflow [-h] -g GENOMES -s SEEDS [-m MODELHOST] -o OUPUT_DIR [-c␣
↪CPU] [-q] [--noorphan] [-p] [-t TARGETS] [--clean] [--pwt-xml] [--target-com-
↪scope]

Run the whole workflow: metabolic network reconstruction, individual and␣
↪community scope analysis and community selection

options:
-h, --help            show this help message and exit
-g GENOMES, --genomes GENOMES
                      annotated genomes directory
-s SEEDS, --seeds SEEDS
                      seeds (growth medium) for metabolic analysis
-m MODELHOST, --modelhost MODELHOST
                      host metabolic model for community analysis
-o OUPUT_DIR, --out OUPUT_DIR
                      output directory path
-c CPU, --cpu CPU     cpu number for multiprocessing
-q, --quiet           quiet mode
--noorphan            use this option to ignore reactions without gene or␣
↪protein association
-p, --padmet          create padmet files
-t TARGETS, --targets TARGETS
                      Optional targets for metabolic analysis, if not used␣
↪metage2metabo will use the addedvalue of the community
--clean               clean PGDBs if already present
--pwt-xml             use this option to use Pathway Tools xml (incompatible␣
↪with -p)
--target-com-scope    Instead of the addedvalue, use the community scope as␣
↪targets for mincom.
```

m2m metacom

```
usage: m2m metacom [-h] -n NETWORKS_DIR -s SEEDS [-m MODELHOST] -o OUPUT_DIR [-
↪t TARGETS] [-q] [-c CPU] [--target-com-scope]

Run the whole metabolism community analysis: individual and community scope␣
↪analysis and community selection

options:
-h, --help            show this help message and exit
-n NETWORKS_DIR, --networksdir NETWORKS_DIR
                      metabolic networks directory
-s SEEDS, --seeds SEEDS
                      seeds (growth medium) for metabolic analysis
-m MODELHOST, --modelhost MODELHOST
                      host metabolic model for community analysis
-o OUPUT_DIR, --out OUPUT_DIR
                      output directory path
```

(continues on next page)

(continued from previous page)

```
-t TARGETS, --targets TARGETS
                        Optional targets for metabolic analysis, if not used␣
↪metage2metabo will use the addedvalue of the community
-q, --quiet           quiet mode
-c CPU, --cpu CPU     cpu number for multiprocessing
--target-com-scope    Instead of the addedvalue, use the community scope as␣
↪targets for mincom.
```

m2m seeds

```
usage: m2m seeds [-h] -o OUPUT_DIR [-q] --metabolites METABOLITES

Create a SBML file starting for a simple text file with metabolic compounds␣
↪identifiers

options:
-h, --help            show this help message and exit
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
-q, --quiet           quiet mode
--metabolites METABOLITES
                        metabolites file: one per line, encoded (XXX as in
↪<species id="XXXX" .../> of SBML files)
```

m2m test

```
usage: m2m test [-h] [-q] [-c CPU] -o OUPUT_DIR

Test the whole workflow on a data sample

options:
-h, --help            show this help message and exit
-q, --quiet           quiet mode
-c CPU, --cpu CPU     cpu number for multiprocessing
-o OUPUT_DIR, --out OUPUT_DIR
                        output directory path
```

2.3 m2m Inputs

2.3.1 Genomes or metabolic networks

If you have annotated genomes and no metabolic networks, you can reconstruct draft metabolic networks using `m2m` with the `m2m workflow` or the `m2m recon` commands. The genomes are expected to be in GenBank format (`.gbk` or `.gbff`) and the input is the directory containing subdirectories with the corresponding files. In addition, the name of the directory must match the name of the genbank file. Example:

```
input_folder
├── organism_1
│   └── organism_1.gbff
├── organism_2
│   └── organism_2.gbff
├── organism_3
│   └── organism_3.gbff
..
└── organism_n
    └── organism_n.gbff
```

If you already have metabolic networks, you can analyse them with the command `m2m metacom` that will perform the whole workflow except for the metabolic reconstruction part. The metabolic networks must be in SBML format. The input is a folder containing multiples SBML files. Example:

```
input_folder
├── organism_1.sbml
├── organism_2.sbml
├── organism_3.sbml
..
└── organism_n.sbml
```

2.3.2 Seeds

Seeds are a set of metabolites that define the nutritional conditions of the community. These metabolites can be components of the growth medium or co-factors of complex cycles (known as currency metabolites). Seeds must be in SBML format.

For example, this is a list of common currency metabolites (from Kim et al. 2015):

```
* proton
* water
* oxygen molecule
* NADP+
* NADPH
* ATP
* diphosphate
* carbon dioxide
* phosphate
* ADP
* coA
* UDP
* NAD+
* NADH
* AMP
* ammonia
* hydrogen peroxide
* oxidized electron acceptor
* reduced electron acceptor
* 3-5-ADP
* GDP
* carbon monoxide
```

(continues on next page)

(continued from previous page)

```
* GTP
* FAD
```

The seed metabolites can be found in the literature. They can be also found from metabolic databases, for example metaboltites from nutrition in the VMH or growth media from MetaCyc (in this latter case, one should used the metabolites in the `Constituents` column).

Once a list of metabolites has been designed, these metabolites must be converted to a list of IDs of the metabolic database corresponding to your metabolic networks. For example, in the VMH seeds ethanol is `etoh`. In the MetaCyc database, the ID of `ethanol` is `ETOH`. Then the ID must be checked with the ID in the SBML files of the metabolic networks. In this example `ETOH` is associated to `M_ETOH_c` in the SBML file in the species field (`<species id="M_ETOH_c" name="ETOH" compartment="c"/>`). The `M_` corresponds to metabolite (another possibility for this prefix is the `R_` for reaction). And `_c` corresponds to the cytosol compartment.

2.3.3 Targets (Optional)

Targets are a set of metabolites whose producibility by the community will be checked and for which minimal communities will be computed. It is possible to give targets as a SBML file to Metage2Metabo with any of the commands `m2m cscope`, `m2m mincom` and `m2m metacom`. Adding targets will replace the metabolites from the cooperation potential (the `addedvalue`) as a metabolic objective whether these metabolites are producible either by individual organisms (with `iscope`), by the community (with `cscope`) and compute minimal communities producing these metabolites (with `mincom`).

The same conversion method than the one for the seeds is needed to create a useful targets file. In addition, one can customise the set of predefined targets from the cooperation potential by modifying the file `community_analysis/targets.sbml` and setting this file as targets with the `-t` argument.

2.3.4 Bibliography

Kim, T., Dreher, K., Nilo-Poyanco, R., Lee, I., Fiehn, O., Lange, B. M., Nikolau, B. J., Sumner, L., Welti, R., Wurtele, E. S., & Rhee, S. Y. (2015). Patterns of metabolite changes identified from large-scale gene perturbations in Arabidopsis using a genome-scale metabolic network. Plant physiology, 167(4), 1685–1698. https://doi.org/10.1104/pp.114.252361

2.4 m2m Tutorial

Test data is avaible in the Github repository. It contains enough data to run the different subcommands.

2.4.1 Outputs and logs

By default, `m2m` writes into the console (stdout) and into a log file located at the root of the results directory and named after the subcommand that was executed. The option `-q` can be given to any `m2m` subcommand to write in the log file and to stdout only the warnings, errors and critical issues, together with the path to the log file.

2.4.2 m2m recon

`m2m recon` runs metabolic network reconstruction for all annotated genomes, using Pathway Tools. It can be done with multiple CPUs, in which case the number of allocated/available CPU has to be given as a optional argument.

It uses the following mandatory inputs (run `m2m recon --help` for optional arguments):

-g directory	directory of annotated genomes
-o directory	output directory for results

Optional arguments:

-c int	number of CPU for multi-processing
--clean	option to rerun every reconstruction even if found in ptools-local
--noorphan	ignore the reactions without gene or protein association in final metabolic networks
-p	create padmet files from PGDB
-l int	specify the level for the sbml to be created
-q	quiet mode
--pwt-xml	extract xml from Pathway Tools instead of using padmet to create sbml

The input genomic data has to follow a strict structure:

```
input_folder
    ├── organism_1
    │   └── organism_1.gbk
    ├── organism_2
    │   └── organism_2.gbk
    ├── organism_3
    │   └── organism_3.gbk
    ..
    └── organism_n
        └── organism_n.gbk
```

This structure can be observed in the *workflow_genomes.tar.gz* file in the Github repository.

By extracting this file, you will find the

```
workflow_genomes
    ├── GCA_003433665
    │   └── GCA_003433665.gbk
    ├── GCA_003433675
    │   └── GCA_003433675.gbk
```

In addition, the genbank files (`.gbk` or `.gbff`) also follow the strict requirements of Pathway Tools. Please go check the documentation of mpwt for more details, especially if you get errors from Pathway Tools at the `recon` step.

```
m2m recon -g workflow_genomes -o output_directory -c cpu_number [--clean] [--orphan] [-
↪p] [-l sbml_level] [--pwt-xml]
```

- standard output

```
##################################################
#                                                #
#       Metabolic network reconstruction         #
#                                                #
##################################################

######### Running metabolic network reconstruction with Pathway Tools #####
↪####
---------- Launching mpwt ----------
|Input Check|GCA_003433665| No missing files
|PathoLogic|GCA_003433665| pathway-tools -no-web-cel-overview -no-cel-
↪overview -no-patch-download -disable-metadata-saving -nologfile -patho␣
↪workflow_genomes/GCA_003433665 -dump-flat-files-biopax
|Input Check|GCA_003433675| No missing files
|PathoLogic|GCA_003433675| pathway-tools -no-web-cel-overview -no-cel-
↪overview -no-patch-download -disable-metadata-saving -nologfile -patho␣
↪workflow_genomes/GCA_003433675 -dump-flat-files-biopax
|Output Check|workflow_genomes/GCA_003433675| 23 out of 23 dat files␣
↪created.
|Moving output files|GCA_003433675|
|Output Check|workflow_genomes/GCA_003433665| 23 out of 23 dat files␣
↪created.
|Moving output files|GCA_003433665|
|Output Check| 2 on 2 builds have passed!
-------------- Checking mpwt runs --------------
All runs are successful.
-------------- mpwt has finished in 251.79s! Thank you for using it. ------
↪--------
######### Creating SBML files #########
######### Stats GSMN reconstruction #########
Number of genomes: 2
Number of reactions in all GSMN: 2432
Number of compounds in all GSMN: 2387
Average reactions per GSMN: 1743.00(+/- 780.65)
Average compounds per GSMN: 1776.00(+/- 695.79)
Average genes per GSMN: 932.00(+/- 504.87)
Percentage of reactions associated with genes: 67.95(+/- 4.90)
--- Recon runtime 256.75 seconds ---

PGDB created in output_directory/pgdb
SBML files created in output_directory/sbml
--- Total runtime 256.75 seconds ---
--- Logs written in output_directory/m2m_recon.log ---


The output shows that PGDB are created with Pathway Tools. Then the .dat␣
↪files are extracted and used to build SBML files of the metabolic models.
```

- **files outputs**
  - In `output_directory/pgdb`, the .dat files of Pathway Tools. The corresponding SBMLs are in `output_directory/sbml`. The structure of the output directory after this `recon` command is shown below :

```
output_directory/
├── m2m_metadata.json
├── m2m_recon.log
├── pgdb
│   ├── GCA_003433665
│   │   ├── classes.dat
│   │   ├── compound-links.dat
│   │   ├── compounds.dat
│   │   ├── dnabindsites.dat
│   │   ├── enzrxns.dat
│   │   ├── gene-links.dat
│   │   ├── genes.dat
│   │   ├── pathway-links.dat
│   │   ├── pathways.dat
│   │   ├── promoters.dat
│   │   ├── protein-features.dat
│   │   ├── protein-links.dat
│   │   ├── proteins.dat
│   │   ├── protligandcplxes.dat
│   │   ├── pubs.dat
│   │   ├── reaction-links.dat
│   │   ├── reactions.dat
│   │   ├── regulation.dat
│   │   ├── regulons.dat
│   │   ├── rnas.dat
│   │   ├── species.dat
│   │   ├── terminators.dat
│   │   └── transunits.dat
│   └── GCA_003433675
│       ├── classes.dat
│       ├── compound-links.dat
│       ├── compounds.dat
│       ├── dnabindsites.dat
│       ├── enzrxns.dat
│       ├── gene-links.dat
│       ├── genes.dat
│       ├── pathway-links.dat
│       ├── pathways.dat
│       ├── promoters.dat
│       ├── protein-features.dat
│       ├── protein-links.dat
│       ├── proteins.dat
│       ├── protligandcplxes.dat
│       ├── pubs.dat
│       ├── reaction-links.dat
│       ├── reactions.dat
│       ├── regulation.dat
│       ├── regulons.dat
│       ├── rnas.dat
│       ├── species.dat
│       ├── terminators.dat
│       └── transunits.dat
└── recon_stats.tsv
```

(continues on next page)

(continued from previous page)

```
└── sbml
    ├── GCA_003433665.sbml
    └── GCA_003433675.sbml

* Finally, in the input directory, some files are also generated automatically␣
↪by Pathway Tools
::

    recon_data/
    ├── GCA_003433665
    │   ├── dat_creation.lisp
    │   ├── GCA_003433665.gbk
    │   ├── genetic-elements.dat
    │   ├── organism-params.dat
    │   └── pathologic.log
    └── GCA_003433675
        ├── dat_creation.lisp
        └── GCA_003433675.gbk
        ├── genetic-elements.dat
        ├── organism-params.dat
        ├── pathologic.log
```

By using the `--pwt-xml`, m2m will extract the xml files created by MetaFlux from the PGDBs created by Pathway Tools. This will modify the files stored in the `pgdb` folder:

```
output_directory/
├── m2m_recon.log
├── m2m_metadata.json
├── pgdb
│   ├── GCA_003433665.xml
│   └── GCA_003433675.xml
└── recon_stats.tsv
└── sbml
    ├── GCA_003433665.sbml
    └── GCA_003433675.sbml
```

2.4.3 m2m iscope, cscope and addedvalue

The three subcommands require metabolic networks in the SBML format. Some metabolic networks are available as a compressed archive in *metabolic_data*. Uncompress the file and the directory can be fed to the subcommands. These commands also require a seeds file comprising the metabolic compounds available to assess reachability/producibility in the models. This seeds file needs to be in SBML format. You can use the one in the `metabolic_data` directory.

iscope

It uses the following mandatory inputs (run `m2m iscope --help` for optional arguments):

-n directory	directory of metabolic networks, in SBML format
-s file	seeds SBML file
-o directory	output directory for results

Optional argument

-q	quiet mode
-c int	number of CPU for multi-processing

```
m2m iscope -n toy_bact -s seeds_toy.sbml -o output_directory/
```

- standard output

```
################################################
#                                              #
#       Individual metabolic potentials        #
#                                              #
################################################


Individual scopes for all metabolic networks available in output_directory/
↪indiv_scopes/indiv_scopes.json. The scopes have been filtered a way that␣
↪if a seed is in a scope, it means the corresponding species is predicted␣
↪to be able to produce it.

Information regarding the producibility of seeds, and the possible absence␣
↪of seeds in some metabolic networks is stored in output_directory/indiv_
↪scopes/seeds_in_indiv_scopes.json.

17 metabolic models considered.

50 metabolites in core reachable by all organisms (intersection)

...

576 metabolites reachable by individual organisms altogether (union),␣
↪among which 44 metabolites that are also part of the seeds (growth␣
↪medium)

...

Summary:
- intersection of scopes 50
- union of scopes 576
- max metabolites in scopes 422
- min metabolites in scopes 116
- average number of metabolites in scopes 239.06 (+/- 89.51)

Analysis of functional redundancy (producers of all metabolites) is␣
↪computed as a dictionary in output_directory/indiv_scopes/rev_iscope.
```

(continues on next page)

(continued from previous page)

```
↪json and as a matrix in output_directory/indiv_scopes/rev_iscope.tsv.
--- Indiv scopes runtime 21.46 seconds ---

--- Total runtime 21.47 seconds ---
--- Logs written in output_directory/m2m_iscope.log --
```

These results mean that 50 metabolites can be reached by all organisms. When gathering reachable metabolites for all organisms, the union consists of 576 metabolites. Some of the reachable metabolites can also be part of the seeds, meaning that there would be a possibility to renew the reservoir of seed molecules by some species. Finally metrics show the min, max and arithmetic mean number of compounds in all scopes.

- files outputs
  - In `output_directory/indiv_scopes/indiv_scopes.json`: a json file that can be easily loaded as a dictionary (or humanly read as it it) that contains the set of reachable metabolites for each organism.
  - A file expliciting the producibility of seeds `output_directory/indiv_scopes/seeds_in_indiv_scopes.json` is also available: it additionally lists the seeds that are absent from the networks.
  - Two more files present the scopes from the focus of metabolites `output_directory/indiv_scopes/rev_iscopes.json` and a matrix summarising the producibility of molecules by species `output_directory/indiv_scopes/rev_iscopes.tsv`. *rev_iscope.tsv* and *rev_iscope.json* that reverse the information from *indiv_scopes.json*. This means that if org1 produces A and B, org2 produces B and C, *indiv_scopes.json* will describe the following: {'org1': ['A', 'B'], 'org2: ['B', 'C']}. *reverse_scope.json* will contain {'A': ['org1'], 'B': ['org1', 'org2'], 'C': ['org2']}, and *reverse_scope.tsv* will contain the same information as a matrix.
  - Logs are written in `output_directory/m2m_iscope.log` and metadata containing package versions in `output_directory/m2m_metadata.json`.

cscope

It uses the following mandatory inputs (run `m2m cscope --help` for optional arguments):

-n directory	directory of metabolic networks, in SBML format
-s file	seeds SBML file
-t file	targets SBML file
-o directory	output directory for results

Optional arguments:

-m file	host metabolic network SBML file
-t file	Optional targets for metabolic analysis, if not used metage2metabo will use the addedvalue of the community
-q	quiet mode

```
m2m cscope -n toy_bact -s seeds_toy.sbml -o output_directory/
```

- standard output

```
##############################################
#                                            #
#    Metabolic potential of the community    #
#                                            #
##############################################

######### Creating metabolic instance for the whole community #########
Created temporary instance file in ../metage2metabo/test/metabolic_data/
↪output_directory/community_analysis/miscoto_9ihtb055.lp
Running whole-community metabolic scopes...
Community scope for all metabolic networks available in output_directory/
↪community_analysis/comm_scopes.json
Contributions of microbes to community scope available in output_directory/
↪community_analysis/contributions_of_microbes.json.


Number of metabolites producible in community: 698.

Reverse community scopes for all metabolic networks available in output_
↪directory/community_analysis/rev_cscope.json and output_directory/
↪community_analysis/rev_cscope.tsv. They higlight the producibility of␣
↪metabolites by species in the community.

--- Community scope runtime 5.41 seconds ---
...
--- Total runtime 5.42 seconds ---
--- Logs written in output_directory/m2m_cscope.log ---
```

698 metabolites (excluding the seeds) reachable by the whole community/microbiota:

- **files outputs**
  - In addition to the logs at the root of the results directory, a json file with the results is created in `output_directory/community_analysis/comm_scopes.json`. It lists all molecules reachable by the community, taking into account the interactions occurring among community members.
  - A file details the roles of community members in the production of metabolites: which microbes possess the reactions that produce the metabolites. This file is `output_directory/community_analysis/contributions_of_microbes.json`. It also recapitulates the compounds producible by species individually versus in community, and highlights the newly producible compounds in community, per symbiont.
  - As for the individual scopes, the redundancy of metabolite producibility is described in `output_directory/community_analysis/rev_cscope.json` and `output_directory/community_analysis/rev_cscope.tsv`.
  - Logs are written in `output_directory/m2m_cscope.log` and metadata containing package versions in `output_directory/m2m_metadata.json`.

addedvalue

`m2m addedvalue` uses the previously two subcommands to compute the added value of combining metabolisms in the microbiota (i.e. consider metabolic cooperation) with respect to studying individually the metabolism of each organism. It uses the following mandatory inputs (run `m2m addedvalue --help` for optional arguments):

-n directory directory of metabolic networks, in SBML format

-s file seeds SBML file

-o directory output directory for results

Optional arguments:

-m file host metabolic network SBML file

-q quiet mode

```
m2m addedvalue -n toy_bact -s seeds_toy.sbml -o output_directory/
```

- **standard output**

```
################################################
#                                              #
#       Individual metabolic potentials        #
#                                              #
################################################


Individual scopes for all metabolic networks available in output_directory/
↪indiv_scopes/indiv_scopes.json. The scopes have been filtered a way that if␣
↪a seed is in a scope, it means the corresponding species is predicted to be␣
↪able to produce it.

Information regarding the producibility of seeds, and the possible absence of␣
↪seeds in some metabolic networks is stored in output_directory/indiv_scopes/
↪seeds_in_indiv_scopes.json.

17 metabolic models considered.

50 metabolites in core reachable by all organisms (intersection)

...

576 metabolites reachable by individual organisms altogether (union), among␣
↪which 44 metabolites that are also part of the seeds (growth medium)

...

Summary:
- intersection of scopes 50
- union of scopes 576
- max metabolites in scopes 422
- min metabolites in scopes 116
- average number of metabolites in scopes 239.06 (+/- 89.51)

Analysis of functional redundancy (producers of all metabolites) is computed␣
```

(continues on next page)

(continued from previous page)

```
↪as a dictionary in output_directory/indiv_scopes/rev_iscope.json and as a␣
↪matrix in output_directory/indiv_scopes/rev_iscope.tsv.
--- Indiv scopes runtime 21.46 seconds ---

################################################
#                                              #
#    Metabolic potential of the community      #
#                                              #
################################################

######### Creating metabolic instance for the whole community #########
Created temporary instance file in ../metage2metabo/test/metabolic_data/output_
↪directory/community_analysis/miscoto_9ihtb055.lp
Running whole-community metabolic scopes...
Community scope for all metabolic networks available in output_directory/
↪community_analysis/comm_scopes.json
Contributions of microbes to community scope available in output_directory/
↪community_analysis/contributions_of_microbes.json.


Number of metabolites producible in community: 698.

Reverse community scopes for all metabolic networks available in output_
↪directory/community_analysis/rev_cscope.json and output_directory/community_
↪analysis/rev_cscope.tsv. They higlight the producibility of metabolites by␣
↪species in the community.

--- Community scope runtime 5.41 seconds ---
...

################################################
#                                              #
#    Added-value of metabolic interactions     #
#                                              #
################################################


Added value of cooperation over individual metabolism: 122 newly reachable␣
↪metabolites:
...
Added-value of cooperation written in output_directory/community_analysis/
↪addedvalue.json
Target file created with the addedvalue targets in: output_directory/community_
↪analysis/targets.sbml
--- Total runtime 27.74 seconds ---
--- Logs written in output_directory/m2m_addedvalue.log ---
```

As you can see, the individual and community scopes are run again. In addition to the previous outputs, the union of all individual scopes and the community scopes are printed. Finally, the difference between the two sets, that is to say the metabolites that can only be produced collectively (i.e. by at least two bacteria cooperating) is displayed. Here it consists of 119 metabolites.

- **files outputs**

- A targets SBML file is generated. It can be used with `m2m mincom` . Newly producible metabolites are written in a json file. The json files associated to `iscope` and `cscope` are also produced.

```
output_directory/
├── m2m_addedvalue.log
├── m2m_metadata.json
├── community_analysis
│   ├── comm_scopes.json
│   ├── addedvalue.json
│   ├── contributions_of_microbes.json
│   ├── rev_cscope.json
│   ├── rev_cscope.tsv
│   └── targets.sbml
├── indiv_scopes
│   └── indiv_scopes.json
│   └── rev_iscope.json
│   └── rev_iscope.tsv
│   └── seeds_in_indiv_scopes.json
```

Optional: create the seeds SBML file

To create a seeds file starting from a list of metabolic identifiers (matching identifiers of compounds of the organisms metabolic networks), you can use the `m2m seeds` command:

```
m2m seeds --metabolites metabolites_file.txt -o output/directory
```

The resulting seeds file will be created in `output/directory/seeds.sbml`.

An example of structure of the metabolites file is the following:

```
M_AMMONIA_c
M_ZN__43__2_c
M_CARBON__45__DIOXIDE_c
M_OXYGEN__45__MOLECULE_c
```

The resulting SBML will have such a design:

```
<?xml version="1.0" encoding="UTF-8"?>
    <sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
    <model id="metabolites">
        <listOfSpecies>
        <species id="M_AMMONIA_c" name="AMMONIA" compartment="c"/>
        <species id="M_ZN__43__2_c" name="ZN+2" compartment="c"/>
        <species id="M_CARBON__45__DIOXIDE_c" name="CARBON-DIOXIDE" compartment="c"/>
        <species id="M_OXYGEN__45__MOLECULE_c" name="OXYGEN-MOLECULE" compartment="c"/>
        </listOfSpecies>
</model>
</sbml>
```

2.4.4 m2m mincom

`m2m mincom` requires an additional target file that is available in *metabolic_data* or can be generated by `m2m addedvalue` in which case it will be stored in `result_directory/community_analysis/targets.sbml`

It uses the following mandatory inputs (run `m2m mincom --help` for optional arguments):

-n directory	directory of metabolic networks, in SBML format
-s file	seeds SBML file
-t file	targets SBML file
-o directory	output directory for results

Optional arguments:

-m file	host metabolic network SBML file
-q	quiet mode

```
m2m mincom -n toy_bact -s seeds_toy.sbml -t targets_toy.sbml -o output_directory/
```

- **standard output**

```
################################################
#                                              #
#         Minimal community selection          #
#                                              #
################################################

WARNING: The following seeds are among the targets: {'M_MANNITOL_c'}. They␣
↪will not be considered as targets during the computation of minimal␣
↪communities: they will be considered as already reachable according to the␣
↪network expansion definition.

Running minimal community selection
/Users/cfrioux/.pyenv/versions/metage2metabo/lib/python3.10/site-packages/
↪miscoto/encodings/community_soup.lp

In the initial and minimal communities 120 targets are producible and 0 remain␣
↪unproducible.

120 producible targets:
...

0 still unproducible targets:


Minimal communities are available in output_directory/community_analysis/
↪mincom.json

######### One minimal community #########
# One minimal community enabling the producibility of the target metabolites␣
↪given as inputs
Minimal number of bacteria in communities => 13

GCA_003437055
```

(continues on next page)

(continued from previous page)

```
GCA_003437595
GCA_003437295
GCA_003437345
GCA_003437715
GCA_003437815
GCA_003437905
GCA_003437375
GCA_003437195
GCA_003438055
GCA_003437885
GCA_003437665
GCA_003437255
######### Key species: Union of minimal communities #########
# Bacteria occurring in at least one minimal community enabling the␣
↪producibility of the target metabolites given as inputs
Number of key species => 17

GCA_003437055
GCA_003437325
GCA_003437595
GCA_003437345
GCA_003437715
GCA_003437905
GCA_003437945
GCA_003438055
GCA_003437255
GCA_003437295
GCA_003437785
GCA_003437815
GCA_003437175
GCA_003437375
GCA_003437195
GCA_003437885
GCA_003437665
######### Essential symbionts: Intersection of minimal communities #########
# Bacteria occurring in ALL minimal communities enabling the producibility of␣
↪the target metabolites given as inputs
Number of essential symbionts => 12

GCA_003437055
GCA_003437595
GCA_003437295
GCA_003437715
GCA_003437815
GCA_003437905
GCA_003437375
GCA_003437195
GCA_003438055
GCA_003437885
GCA_003437665
GCA_003437255
######### Alternative symbionts: Difference between Union and Intersection ####
```

(continues on next page)

(continued from previous page)

```
↪#####
# Bacteria occurring in at least one minimal community but not all minimal␣
↪communities enabling the producibility of the target metabolites given as␣
↪inputs
Number of alternative symbionts => 5

GCA_003437345
GCA_003437325
GCA_003437945
GCA_003437785
GCA_003437175

--- Mincom runtime 5.61 seconds ---

--- Total runtime 7.72 seconds ---
--- Logs written in output_directory/m2m_mincom.log ---
```

This output gives the result of minimal community selection. It means that for producing the 120 metabolic targets, a minimum of 13 bacteria out of the 17 is required. One example of such minimal community is given. In addition, the whole space of solution is studied. All bacteria (17) occur in at least one minimal community (key species). Finally, the intersection gives the following information: a set of 12 bacteria occurs in each minimal communtity. This means that these 12 bacteria are needed in any case (essential symbionts), and that any of the remaining 5 bacteria (alternative symbionts) can complete the missing function(s).

- **files outputs**
  - As for other commands, a json file with the results is produced in `output_directory/community_analysis/mincom.json`, together with logs at the root of the results directory.

2.4.5 m2m metacom

`m2m metacom` runs all analyses: individual scopes, community scopes, and minimal community selection based on the metabolic added-value of the microbiota.

It uses the following mandatory inputs (run `m2m metacom --help` for optional arguments):

-n directory	directory of metabolic networks, in SBML format
-s file	seeds SBML file
-o directory	output directory for results

Optional arguments:

-m file	host metabolic network SBML file
-t file	Optional targets for metabolic analysis, if not used metage2metabo will use the addedvalue of the community
-q	quiet mode
-c int	number of CPU for multi-processing
--target-com-scope	Instead of the addedvalue, use the community scope as targets for mincom

```
m2m metacom -n metabolic_data/toy_bact -s metabolic_data/seeds_toy.sbml -o output_
↪directory
```

- **standard output**

```
######### Running individual metabolic scopes #########
Individual scopes for all metabolic networks available in output_directory/
↪indiv_scopes/indiv_scopes.json
17 metabolic models considered.

135 metabolites in core reachable by all organisms (intersection)

...

625 metabolites reachable by individual organisms altogether (union), among␣
↪which 93 seeds (growth medium)

...

intersection of scope 135
union of scope 625
max metabolites in scope 477
min metabolites in scope 195
average number of metabolites in scope 308.71 (+/- 82.59)
Analysis of functional redundancy (producers of all metabolites) is computed␣
↪as a dictionary in output_directory/indiv_scopes/rev_iscope.json and as a␣
↪matrix in output_directory/indiv_scopes/rev_iscope.tsv.
--- Indiv scopes runtime 9.77 seconds ---

######### Creating metabolic instance for the whole community #########
Created instance in /shared/programs/metage2metabo/test/output_directory/
↪community_analysis/miscoto_wkdkeazl.lp
Running whole-community metabolic scopes
Community scopes for all metabolic networks available in output_directory/
↪community_analysis/comm_scopes.json
--- Community scope runtime 5.84 seconds ---


Added value of cooperation over individual metabolism: 119 newly reachable␣
↪metabolites:

...


Added-value of cooperation written in output_directory/community_analysis/
↪addedvalue.json

Target file created with the addedvalue targets in: output_directory/community_
↪analysis/targets.sbml
Setting 119 compounds as targets

Running minimal community selection

In the initial and minimal communities 119 targets are producible and 0 remain␣
↪unproducible.
```

(continues on next page)

(continued from previous page)

```
119 producible targets:
...

0 still unproducible targets:


Minimal communities are available in output_directory/community_analysis/
↪mincom.json

######### One minimal community #########
# One minimal community enabling the producibility of the target metabolites␣
↪given as inputs
Minimal number of bacteria in communities => 13

GCA_003437255
GCA_003437885
GCA_003437815
GCA_003437375
GCA_003437295
GCA_003437715
GCA_003437665
GCA_003438055
GCA_003437195
GCA_003437905
GCA_003437595
GCA_003437055
GCA_003437945
######### Key species: Union of minimal communities #########
# Bacteria occurring in at least one minimal community enabling the␣
↪producibility of the target metabolites given as inputs
Number of key species => 17

GCA_003437785
GCA_003437885
GCA_003437055
GCA_003437345
GCA_003437665
GCA_003437195
GCA_003437905
GCA_003437175
GCA_003437595
GCA_003437325
GCA_003437815
GCA_003437375
GCA_003437295
GCA_003437715
GCA_003437255
GCA_003438055
GCA_003437945
######### Essential symbionts: Intersection of minimal communities #########
# Bacteria occurring in ALL minimal communities enabling the producibility of␣
↪the target metabolites given as inputs
```

(continues on next page)

(continued from previous page)

```
Number of essential symbionts => 12

GCA_003437255
GCA_003437885
GCA_003437815
GCA_003437295
GCA_003437375
GCA_003437715
GCA_003437665
GCA_003438055
GCA_003437195
GCA_003437905
GCA_003437595
GCA_003437055
######### Alternative symbionts: Difference between Union and Intersection ####
↪#####
# Bacteria occurring in at least one minimal community but not all minimal␣
↪communities enabling the producibility of the target metabolites given as␣
↪inputs
Number of alternative symbionts => 5

GCA_003437345
GCA_003437945
GCA_003437175
GCA_003437325
GCA_003437785

--- Mincom runtime 4.34 seconds ---

Targets producibility are available at output_directory/producibility_targets.
↪json
--- Total runtime 20.01 seconds ---
--- Logs written in output_directory/m2m_metacom.log ---
```

- **files outputs**
  - Files are created in the output_directory: the logs, json files with the results, targets in SBML.

```
output_directory/
├── m2m_metadata.json
├── m2m_mincom.log
├── producibility_targets.json
├── community_analysis
│   ├── addedvalue.json
│   ├── comm_scopes.json
│   ├── contributions_of_microbes.json
│   ├── mincom.json
│   ├── rev_cscope.json
│   ├── rev_cscope.tsv
│   └── targets.sbml
├── indiv_scopes
│   └── indiv_scopes.json
│   └── rev_iscope.json
```

(continues on next page)

(continued from previous page)

```
    └── rev_iscope.tsv
    └── seeds_in_indiv_scopes.json
```

2.4.6 m2m workflow and m2m test

`m2m workflow` starts from metabolic network reconstruction and runs all analyses: individual scopes, community scopes, and minimal community selection based on the metabolic added-value of the microbiota.

It uses the following mandatory inputs (run `m2m workflow --help` for optional arguments):

-g directory	directory of annotated genomes
-s file	seeds SBML file
-o directory	output directory for results

Optional arguments:

-c int	number of CPU for multi-processing
--clean	option to rerun every reconstruction even if found in ptools-local
--noorphan	ignore the reactions without gene or protein association in final metabolic networks
-p	create padmet files from PGDB
-t file	Optional targets for metabolic analysis, if not used metage2metabo will use the addedvalue of the community
-q	quiet mode
-m file	host metabolic network SBML file
--pwt-xml	extract xml from Pathway Tools instead of using padmet to create sbml
--target-com-scope	Instead of the addedvalue, use the community scope as targets for mincom

You can run the workflow analysis with the two genbanks files available in the Github repository (*workflow_data*). Two genomes are available in the compressed archive *workflow_genomes.tar.gz*. The archive has to be uncompressed before testing.

```
m2m workflow -g workflow_genomes -s seeds_workflow.sbml -o output_directory/
```

Or you can run the test argument (which use the same data): `m2m test`.

Which uses the following mandatory inputs (run `m2m test --help` for optional arguments):

-o directory	output directory path

Optional arguments:

-q	quiet mode
-c int	cpu number for multi-processing

```
m2m test -o output_directory
```

- standard outputs

```
################################################
#                                              #
#       Metabolic network reconstruction       #
#                                              #
################################################

######### Running metabolic network reconstruction with Pathway Tools #####
↪####
---------- Launching mpwt ----------
|Input Check|GCA_003433665| No missing files
|PathoLogic|GCA_003433665| pathway-tools -no-web-cel-overview -no-cel-
↪overview -no-patch-download -disable-metadata-saving -nologfile -patho␣
↪workflow_genomes/GCA_003433665 -dump-flat-files-biopax
|Output Check|workflow_genomes/GCA_003433665| 23 out of 23 dat files␣
↪created.
|Moving output files|GCA_003433665|
|Input Check|GCA_003433675| No missing files
|PathoLogic|GCA_003433675| pathway-tools -no-web-cel-overview -no-cel-
↪overview -no-patch-download -disable-metadata-saving -nologfile -patho␣
↪workflow_genomes/GCA_003433675 -dump-flat-files-biopax
|Output Check|workflow_genomes/GCA_003433675| 23 out of 23 dat files␣
↪created.
|Moving output files|GCA_003433675|
|Output Check| 2 on 2 builds have passed!
-------------- Checking mpwt runs --------------
All runs are successful.
-------------- mpwt has finished in 415.62s! Thank you for using it. ------
↪--------
######### Creating SBML files #########
######### Stats GSMN reconstruction #########
Number of genomes: 2
Number of reactions in all GSMN: 2433
Number of compounds in all GSMN: 2389
Average reactions per GSMN: 1743.50(+/- 781.35)
Average compounds per GSMN: 1777.00(+/- 697.21)
Average genes per GSMN: 932.00(+/- 504.87)
Percentage of reactions associated with genes: 67.96(+/- 4.91)
--- Recon runtime 421.74 seconds ---


################################################
#                                              #
#       Individual metabolic potentials        #
#                                              #
################################################


Individual scopes for all metabolic networks available in output_directory/
↪indiv_scopes/indiv_scopes.json. The scopes have been filtered a way that␣
↪if a seed is in a scope, it means the corresponding species is predicted␣
↪to be able to produce it.

Information regarding the producibility of seeds, and the possible absence␣
```

(continues on next page)

(continued from previous page)

```
↪of seeds in some metabolic networks is stored in output_directory/indiv_
↪scopes/seeds_in_indiv_scopes.json.

2 metabolic models considered.

143 metabolites in core reachable by all organisms (intersection)

...

334 metabolites reachable by individual organisms altogether (union),␣
↪among which 12 metabolites that are also part of the seeds (growth␣
↪medium)

...

Summary:
- intersection of scopes 143
- union of scopes 334
- max metabolites in scopes 333
- min metabolites in scopes 144
- average number of metabolites in scopes 238.50 (+/- 133.64)

Analysis of functional redundancy (producers of all metabolites) is␣
↪computed as a dictionary in output_directory/indiv_scopes/rev_iscope.
↪json and as a matrix in output_directory/indiv_scopes/rev_iscope.tsv.
--- Indiv scopes runtime 0.83 seconds ---


################################################
#                                              #
#    Metabolic potential of the community      #
#                                              #
################################################

######### Creating metabolic instance for the whole community #########
Created temporary instance file in /shared/Softwares/git/metage2metabo/
↪metage2metabo/workflow_data/output_directory/community_analysis/miscoto_
↪bdg_458k.lp
Running whole-community metabolic scopes...
Community scope for all metabolic networks available in output_directory/
↪community_analysis/comm_scopes.json
Contributions of microbes to community scope available in output_directory/
↪community_analysis/contributions_of_microbes.json.


Number of metabolites producible in community: 357.

Reverse community scopes for all metabolic networks available in output_
↪directory/community_analysis/rev_cscope.json and output_directory/
↪community_analysis/rev_cscope.tsv. They higlight the producibility of␣
↪metabolites by species in the community.
```

(continues on next page)

(continued from previous page)

```
--- Community scope runtime 0.72 seconds ---


################################################
#                                              #
#    Added-value of metabolic interactions     #
#                                              #
################################################


Added value of cooperation over individual metabolism: 23 newly reachable␣
↪metabolites:

...


Added-value of cooperation written in output_directory/community_analysis/
↪addedvalue.json

Use the addedvalue as targets.

Target file created with the addedvalue targets in: output_directory/
↪community_analysis/targets.sbml
Setting 23 compounds as targets.


################################################
#                                              #
#         Minimal community selection          #
#                                              #
################################################

Running minimal community selection
/shared/Softwares/git/miscoto/miscoto/encodings/community_soup.lp

In the initial and minimal communities 23 targets are producible and 0␣
↪remain unproducible.

23 producible targets:
...

0 still unproducible targets:


Minimal communities are available in output_directory/community_analysis/
↪mincom.json

######### One minimal community #########
# One minimal community enabling the producibility of the target␣
↪metabolites given as inputs
Minimal number of bacteria in communities => 2
```

(continues on next page)

(continued from previous page)

```
...

######### Key species: Union of minimal communities #########
# Bacteria occurring in at least one minimal community enabling the␣
↪producibility of the target metabolites given as inputs
Number of key species => 2

...

######### Essential symbionts: Intersection of minimal communities ########
↪#
# Bacteria occurring in ALL minimal communities enabling the producibility␣
↪of the target metabolites given as inputs
Number of essential symbionts => 2

...

######### Alternative symbionts: Difference between Union and Intersection
↪#########
# Bacteria occurring in at least one minimal community but not all minimal␣
↪communities enabling the producibility of the target metabolites given␣
↪as inputs
Number of alternative symbionts => 0



--- Mincom runtime 0.88 seconds ---

Targets producibility are available at output_directory/producibility_
↪targets.json
--- Total runtime 424.18 seconds ---
--- Logs written in output_directory/m2m_workflow.log ---
```

- **files outputs**
  - Numerous files are created in the *output_directory*, including the logs at the root of the results directory.

```
output_directory/
├── m2m_metadata.json
├── m2m_workflow.log
├── producibility_targets.json
├── community_analysis
│   ├── addedvalue.json
│   ├── comm_scopes.json
│   ├── contributions_of_microbes.json
│   ├── mincom.json
│   ├── rev_cscope.json
│   ├── rev_cscope.tsv
│   └── targets.sbml
├── indiv_scopes
│   └── indiv_scopes.json
│   └── rev_iscope.json
```

(continues on next page)

(continued from previous page)

```
    └── rev_iscope.tsv
    └── seeds_in_indiv_scopes.json
├── padmet
    ├── GCA_003433665.padmet
    └── GCA_003433675.padmet
├── pgdb
    ├── GCA_003433665
        ├── classes.dat
        ├── compound-links.dat
        ├── compounds.dat
        ├── dnabindsites.dat
        ├── enzrxns.dat
        ├── gene-links.dat
        ├── genes.dat
        ├── pathway-links.dat
        ├── pathways.dat
        ├── promoters.dat
        ├── protein-features.dat
        ├── protein-links.dat
        ├── proteins.dat
        ├── protligandcplxes.dat
        ├── pubs.dat
        ├── reaction-links.dat
        ├── reactions.dat
        ├── regulation.dat
        ├── regulons.dat
        ├── rnas.dat
        ├── species.dat
        ├── terminators.dat
        └── transunits.dat
    └── GCA_003433675
        ├── classes.dat
        ├── compound-links.dat
        ├── compounds.dat
        ├── dnabindsites.dat
        ├── enzrxns.dat
        ├── gene-links.dat
        ├── genes.dat
        ├── pathway-links.dat
        ├── pathways.dat
        ├── promoters.dat
        ├── protein-features.dat
        ├── protein-links.dat
        ├── proteins.dat
        ├── protligandcplxes.dat
        ├── pubs.dat
        ├── reaction-links.dat
        ├── reactions.dat
        ├── regulation.dat
        ├── regulons.dat
        ├── rnas.dat
        ├── species.dat
```

(continues on next page)

(continued from previous page)

```
│       ├── terminators.dat
│       └── transunits.dat
└── recon_stats.tsv
└── sbml
    ├── GCA_003433665.sbml
    └── GCA_003433675.sbml
```

These files are the same as the ones presented in the previous commands: metabolic networks reconstructions (Pathway Tools data, SBML), individual and collective scopes, minimal community selection.

`m2m metacom` runs the whole workflow except the reconstruction of metabolic networks. We advise to use this command to explore the metabolism of the microbial community when you already have metabolic networks.

2.4.7 Including a host in the picture

It is possible to consider a host in addition to the microbiota for the `workflow`, `cscope` and `mincom` commands. **What does it change?**

First note that adding the host in the SBML repository will enable you to get the individual scope for the host. Another solution is to directly use `menescope` from the MeneTools Python package on which m2m relies, and that can be used as a standalone tool.

Then back to the effect of the host in the other commands.

- For `cscope` and `addedvalue`, the host metabolism will be taken into account. That is to say that it will be considered as a member of the community. Among the newly producible targets, some will be exclusive to the host metabolism. This is not displayed in the standard output of the software but can be retrieved in the json file output under the *"comhost_scope"* key of the dictionary.
- For `mincom`, the host will always be considered in the community. This means that the selected bacteria need to be associated to the host in order to ensure the producibility of all the targets. Therefore, if the minimal community computed for 10 targets is of 3 bacteria and that a host was provided, it means that the host + these three bacteria can produce the 10 targets.

More generally, for more information and analysis on the usage of hosts in addition to the microbiota, we refer the interested user to the Miscoto Python package, on which m2m relies. Miscoto can be used as a standalone package for such analyses, with additional options, such as the identification of putative exchanges among the minimal communities.

2.4.8 More information

Take a look at the complete tutorial in the Github repository

2.5 m2m Outputs

- m2m steps will create multiple results files in an output folder:

```
output_directory/
├── m2m_metadata.json
├── m2m_command.log
├── pgdb
│   ├── species_1
```

(continues on next page)

(continued from previous page)

```
│   │   ├── classes.dat
│   │   ├── compound-links.dat
│   │   ├── compounds.dat
│   │   ├── dnabindsites.dat
│   │   ├── enzrxns.dat
│   │   ├── gene-links.dat
│   │   ├── genes.dat
│   │   ├── pathway-links.dat
│   │   ├── pathways.dat
│   │   ├── promoters.dat
│   │   ├── protein-features.dat
│   │   ├── protein-links.dat
│   │   ├── proteins.dat
│   │   ├── protligandcplxes.dat
│   │   ├── pubs.dat
│   │   ├── reaction-links.dat
│   │   ├── reactions.dat
│   │   ├── regulation.dat
│   │   ├── regulons.dat
│   │   ├── rnas.dat
│   │   ├── species.dat
│   │   ├── terminators.dat
│   │   └── transunits.dat
│   └── ...
└── sbml
    ├── species_1.sbml
    └── ...
├── padmet
│   ├── species_1.padmet
│   └── ...
└── recon_stats.tsv
├── indiv_scopes
│   └── indiv_produced_seeds.json
│   └── indiv_scopes.json
│   └── rev_iscope.json
│   └── rev_iscope.tsv
├── community_analysis
│   ├── addedvalue.json
│   ├── comm_scopes.json
│   ├── contributions_of_microbes.json
│   ├── mincom.json
│   ├── rev_cscope.json
│   ├── rev_cscope.tsv
│   └── targets.sbml
├── producibility_targets.json
```

By using the `--pwt-xml` option, m2m will use the xml files from Pathway Tools and there will be no .dat files.

```
output_directory/
├── m2m_metadata.json
├── m2m_command.log
├── pgdb
```

(continues on next page)

(continued from previous page)

```
│   ├── species_1.xml
│   └── ...
└── sbml
    ├── species_1.sbml
    └── ...
└── recon_stats.tsv
├── indiv_scopes
    └── indiv_produced_seeds.json
    └── indiv_scopes.json
    └── rev_iscope.json
    └── rev_iscope.tsv
├── community_analysis
    ├── addedvalue.json
    ├── comm_scopes.json
    ├── contributions_of_microbes.json
    ├── mincom.json
    ├── rev_cscope.json
    ├── rev_cscope.tsv
    └── targets.sbml
├── producibility_targets.json
```

2.5.1 m2m_metadata.json

A json file summarising metadata of Metage2Metabo:

- `m2m_args`: input arguments given to Metage2Metabo such as the command used, the path to genomes or networks, etc.
- `tool_dependencies`: Python version, versions of Metage2Metabo and its dependencies and Clingo version. When using `recon` or `workflow` commands it will also contains Pathway Tools version.
- `duration`: duraiton of the un of Metage2Metabo in seconds.

2.5.2 m2m_command.log

A log file named after the command used. For example, if you launch `m2m metacom`, the file will be named `m2m_metacom.log`.

This file contains all the logs return by m2m.

2.5.3 pgdb folder

If you use the reconstruction (with `m2m recon` or `m2m workflow`), the draft metabolic network inferred by Pathway Tools for each of your species will be stored in this `pgdb` folder.

For each genome, if no errors occur, there will be a folder named after the genome folder name containing the Pathway-Genome Database (PGDB). The PGDB is stored using attribute-values flat files (the .dat extension file).

By using the `--pwt-xml`, m2m will extract the xml created by MetaFlux (a module of Pathway Tools) instead of extracting the attribute-values flat .dat files.

2.5.4 sbml

After Pathway Tools metabolic network reconstruction, m2m will create SBML files from the PGDB attribute-values files. All the sbml are stored in this `sbml` folder.

There is one sbml for each genome given as input to m2m. These metabolic network in SBML are the input of the non-reconstruction step of m2m.

2.5.5 padmet

If you use the `-p` argument with the reconstruction, m2m will also create padmet file, a format used to store metabolic network information and metadata.

One padmet file is created for each genome in the PGDB folder.

2.5.6 recon_stats.tsv

After the reconstruction, m2m will summary the information of the draft metabolic networks in this file.

It will contain the number of genes, reactions, compounds and pathways in each metabolic networks.

2.5.7 indiv_scopes

The indiv_scopes folder is created after the individual scopes step (in `m2m worfklow`, `m2m metacom` or `m2m iscope`). This step uses a folder containing multiples metabolic network in SBMLs and a seed file (also in SBML).

The results are stored in a json file named `indiv_scopes.json`. The keys in this file are each metabolic network and the values are the compounds that can be produced individually by the metabolic network.

Also it can occur that seeds are produbile by individual organisms, in this case they will be listed in `indiv_produced_seeds.json`. The keys in this file are each metabolic network and the values are the seeds that can be produced individually by the metabolic network.

The reverse of the previous iscope dictionary is stored in two files `rev_iscope.json` and `rev_iscope.tsv`. The latter file is a matrix with compounds as column header and species in row. For each compounds, we have the producibility of the compounds by the species (0 not producible and 1 producible).

2.5.8 community_analysis

The community_analysis folder stores all the results involving the community analysis (`m2m worfklow`, `m2m metacom`, `m2m cscope`, `m2m addedvalue` or `m2m mincom`).

comm_scopes.json

First step of the community analysis after the individual production analysis, the community scopes (called by `m2m worfklow`, `m2m metacom` or `m2m cscope`) shows the compounds producible by the input metabolic networks with cooperation.

The results are stored in a json with 8 keys:

- `host_prodtargets`: if a host is given as input, contains the targets producible by the host.
- `host_unprodtargets`: if a host is given as input, contains the targets not producible by the host.
- `host_scope`: if a host is given as input, contains all the compounds producible by the host.

- com_prodtargets: the targets producible by the community.
- com_unprodtargets: the targets not producible by the community.
- comhost_scope: all the compounds producible by the host + the community.
- com_scope: all the compounds producible by the community.
- targets_producers: for each target, the list of organisms able to produce this target. It is empty if you use m2m worfklow or m2m metacom without targets because cscope needs target to find the targets_producers.

rev_cscope.json and rev_cscope.tsv

The reverse of the comm_scopes.json scope keys are stored in two files rev_cscope.json and rev_cscope.tsv. It shows for each metabolites, which species in the community can produce it. The latter file is a matrix with compounds as column header and species in row. For each compounds, we have the producibility of the compounds by the species (0 not producible and 1 producible).

addedvalue.json

After the individual scopes and the community scopes, the addedvalue (m2m worfklow, m2m metacom, m2m addedvalue), extracts the compounds that are producible by the community but not by individual organism.

The result are stored in a json file with one key addedvalue which enumerates all the compounds producible by the community but not by the individual organism.

targets.sbml

After the addedvalue (m2m worfklow, m2m metacom, m2m addedvalue), all the compounds that have been found by this step are stored in this sbml file. It is used as the targets file for the following step.

mincom.json

Using the addedvalue or targets given by the user, the mincom step (m2m worfklow, m2m metacom or m2m mincom) will search for the minimal community that can produce these compounds.

The results are stored in a json with 17 keys:

- bacteria: organisms in the optimal solution.
- still_unprod: compounds unproducible by the community.
- newly_prod: compounds producible by the community.
- union_bacteria: organisms from all the minimal communities.
- inter_bacteria: organisms from the intersection of all the minimal communities.
- one_model: results of the optimal solution.
- exchanged, union_exchanged and inter_exchanged: the exchanged compounds by the community, this step needs a lot of resources so it is not used in m2m. If you want to use it, use miscoto with the minexch option.
- key_species: organisms from all the minimal communities.
- essential_symbionts: organisms in the intersection of all the minimal communities. They are occuring in all minimal solution.
- alternative_symbionts: organisms appearing in at least one minimal community but not in all.

- `score_optimum_inter`: the optimum score found for the intersection, it corresponds to the number of organism in the minimal community.
- `score_optimum_union`: the optimum score found for the union, it corresponds to the number of organism in the minimal community.
- `inter_targetsproducers`: the organism that have the final reaction to produce the target in the intersection. It is a dictionary, with each target as key and the organism producing these targets as value.
- `union_targetsproducers`: the organism that have the final reaction to produce the target in the union. It is a dictionary, with each target as key and the organism producing these targets as value.
- `one_model_targetsproducers`: the organism that have the final reaction to produce the target in the optimal solution. It is a dictionary, with each target as key and the organism producing these targets as value.

contributions_of_microbes.json

A json file detailing the role of community members in the production of metabolites: which organisms have the reactions that produce the metabolites.

It contains one key per microbe in the community with several subkeys:

- `produced_alone`: metabolties that can be produced by the organism alone.
- `community_metabolic_gain`: metabolties that can be newly produced by the organism with the help of the community.
- `produced_in_community`: metabolties that can be produced by the organism when it is in the community (it is composed of `community_metabolic_gain` and `produced_alone`).

2.5.9 producibility_targets.json

After all these previous step, m2m (`m2m worfklow` or `m2m metacom`) will create this json which summarizes the producibility of each targets (either given by the user or from the addedvalue).

This json contains 12 keys:

- `producible`: the producible compounds by the community.
- `unproducible`: the unproducible compounds by the community.
- `indiv_producible`: the compounds producible by individual organisms.
- `individual_producers`: for each targets the individual organisms that can produce them.
- `com_only_producers`: the organism that have the final reaction to produce the target but needs other organisms to produces the previous compounds needed by this final reaction. It is a dictionary, with each target as key and the organism producing these targets as value.
- `mincom_producible`: the compounds producible by the minimal community.
- `key_species`: organism from all the minimal communities.
- `essential_symbionts`: organisms in the intersection of all the minimal communities. They are occuring in all minimal solution.
- `alternative_symbionts`: organisms appearing in at least one minimal community but not in all.
- `mincom_inter_producers`: the organism that have the final reaction to produce the target in the intersection. It is a dictionary, with each target as key and the organism producing these targets as value.

- `mincom_union_producers`: the organism that have the final reaction to produce the target in the union. It is a dictionary, with each target as key and the organism producing these targets as value.
- `mincom_optsol_producers`: the organism that have the final reaction to produce the target in the optimal solution. It is a dictionary, with each target as key and the organism producing these targets as value.

2.6 Supplementary information

2.6.1 Scope criterion of graph-based producibility

The criterion used to assess producibility in m2m is graph-based. It relies on the definition of scope introduced by [Eb2004].

More precisely, it uses seeds in addition to the metabolic model itself, that is to say the information on which compounds are available to initiate producibility. The seeds often consists in metabolites found in the growth medium.

Starting from the seeds, the sub-network that is reachable, also called *scope* is expanded until it reaches a fixed point. A metabolite is considered producible or reachable from the seeds if it belongs to the scope. Similarly, a reaction is considered activable if all of its reactants are in the scope, that is to say are reachable. It is important to notice the conjunction in this rule : *all* the reactants need to be in the scope.

Therefore, the scope is computed following these two conditions:

- initiation condition: a metabolite is in the scope if it belongs to the seeds.
- recursive condition: a metabolite is in the scope if it is the product of a reaction whose reactants are in the scopes (all of them).

The computation of the scope in m2m is performed with logic programming: Answer Set Programming.

In the example above, a small metabolic network consists of 5 reactions (squares) and 8 metabolites (circles). *A* and *B* are the seeds. The yellow compounds or reactions are respectively reachable or activable. In particular, note that *H* cannot be produced because one of the reactants of *R5* is not producible.

2.7 m2m's API

2.7.1 Workflow

m2m workflow

metage2metabo.m2m.m2m_workflow.**add_targets_to_instance**(*instancefile*, *output_dir*, *target_set*)

Add targets to the ASP community instance file.

Parameters

- **instancefile** (*str*) – instance filepath
- **output_dir** (*str*) – directory for results
- **target_set** (*set*) – targets to be added

Returns

new instance filepath

Return type

str

metage2metabo.m2m.m2m_workflow.**metacom_analysis**(*sbml_dir*, *out_dir*, *seeds*, *host_mn*, *targets_file*, *cpu_number=1*, *target_com_scope=None*)

Run the metabolism community analysis part of m2m.

Parameters

- **sbml_dir** (*str*) – sbml input directory
- **out_dir** (*str*) – results directory
- **seeds** (*str*) – seeds file
- **host_mn** (*str*) – metabolic network file for host
- **targets_file** (*str*) – targets file
- **cpu_number** (*int*) – number of CPU to use for multiprocessing
- **target_com_scope** (*bool*) – if True, will use all metabolties in com_scope as targets for minimal community predictions.

metage2metabo.m2m.m2m_workflow.**run_workflow**(*inp_dir*, *out_dir*, *nb_cpu*, *clean*, *seeds*, *noorphan_bool*, *padmet_bool*, *host_mn*, *targets_file*, *use_pwt_xml*, *target_com_scope=None*)

Run the whole m2m workflow.

Parameters

- **inp_dir** (*str*) – genomes directory
- **out_dir** (*str*) – results directory
- **nb_cpu** (*int*) – cpu number for multi-processing
- **clean** (*bool*) – clean PGDB and re-run them
- **seeds** (*str*) – seeds file
- **noorphan_bool** (*bool*) – ignores orphan reactions if True
- **padmet_bool** (*bool*) – creates padmet files if True
- **host_mn** (*str*) – metabolic network file for host
- **targets_file** (*str*) – targets file
- **use_pwt_xml** (*bool*) – use Pathway Tools XML instead of creating them with padmet
- **target_com_scope** (*bool*) – if True, will use all metabolties in com_scope as targets for minimal community predictions.

metage2metabo.m2m.m2m_workflow.**targets_producibility**(*m2m_out_dir*, *union_targets_iscope*, *targets_cscope*, *addedvalue_targets*, *user_targets=None*, *target_com_scope=None*)

Create a json summarizing the producibility of the targets (either the addedvalue or the user provided targets)

Parameters

- **m2m_out_dir** (*str*) – M2M results directory
- **union_targets_iscope** (*list*) – targets producible by indiviual
- **targets_cscope** (*list*) – targets producible by community
- **addedvalue_targets** (*list*) – targets produbed by the community and not by individual

- **user_targets** (*list*) – targets provided by the user
- **target_com_scope** (*bool*) – if True, will use all metabolties in com_scope as targets for minimal community predictions.

Reconstruction

metage2metabo.m2m.reconstruction.**analyze_recon**(*sbml_folder*, *output_stat_file*, *padmet_folder=None*, *padmet_bool=None*, *nb_cpu=1*)

Analyze the sbml and/or the padmet files after metabolic network reconstruction. And write the result in a file.

Parameters

- **sbml_folder** (*str*) – directory of SBML files
- **output_stat_file** (*str*) – path to output stat file
- **padmet_folder** (*str*) – directory of PADMET files
- **padmet_bool** (*bool*) – use or not the padmet files
- **nb_cpu** (*int*) – number of CPU to use

metage2metabo.m2m.reconstruction.**create_padmet_stat**(*species_name*, *padmet_file*)

Extract reactions/pathways/compounds/genes from a padmet file.

Parameters

- **species_name** (*str*) – species names
- **padmet_file** (*str*) – path to a padmet file

Returns

list: [species name, list of genes, list of reactions, list of reactions associated with genes, list of compounds, list of pathways]

metage2metabo.m2m.reconstruction.**create_sbml_stat**(*species_name*, *sbml_file*)

Extract reactions/pathways/compounds/genes from a sbml file.

Parameters

- **species_name** (*str*) – species names
- **sbml_file** (*str*) – path to a sbml file

Returns

list: [species name, list of genes, list of reactions, list of reactions associated with genes, list of compounds]

metage2metabo.m2m.reconstruction.**genomes_to_pgdb**(*genomes_dir*, *output_dir*, *cpu*, *clean*, *use_pwt_xml*)

Run Pathway Tools on each genome of the repository

Parameters

- **genomes_dir** (*str*) – genome repository
- **output_dir** (*str*) – output repository
- **cpu** (*int*) – number of CPUs to use
- **clean** (*bool*) – delete PGDBs in ptools-local coresponding to the input data
- **use_pwt_xml** (*bool*) – use Pathway Tools XML instead of creating them with padmet

Returns
pgdb repository

Return type
pgdb_dir (str)

metage2metabo.m2m.reconstruction.**mean_sd_data**(*datas*)

Compute the mean and standard deviation from a list.

Parameters
datas (*list*) – list of integer/float

Returns
mean_data (float): mean of the list sd_data (flaot): standard deviation of the lsit

metage2metabo.m2m.reconstruction.**recon**(*inp_dir*, *out_dir*, *noorphan_bool*, *padmet_bool*, *sbml_level*, *nb_cpu*, *clean*, *use_pwt_xml*)

Run metabolic network reconstruction with Pathway Tools and get SBMLs.

Parameters

- **inp_dir** (*str*) – genomes directory
- **out_dir** (*str*) – results directory
- **noorphan_bool** (*bool*) – ignores orphan reactions if True
- **padmet_bool** (*bool*) – creates padmet files if True
- **sbml_level** (*str*) – SBML level (2 or 3)
- **nb_cpu** (*int*) – number of CPU for multiprocessing
- **clean** (*bool*) – re-run metabolic reconstructions that are already available if found
- **use_pwt_xml** (*bool*) – use Pathway Tools XML instead of creating them with padmet

Returns
PGDB directory (str), SBML directory (str)

Return type
tuple

metage2metabo.m2m.reconstruction.**update_pathway_tools_xml**(*input_sbml*, *output_sbml*)

Update XML from Pathway Tools by adding a 'M_' prefix to avoid issue, when using metaboltie IDs.

Parameters

- **input_sbml** (*str*) – path to xml input file
- **output_sbml** (*str*) – path to xml output file

Individual scope

metage2metabo.m2m.individual_scope.**analyze_indiv_scope**(*scope_dict*, *seeds_status_dict*, *seeds*)

Analyze the output of Menescope, stored in two dictionaries

Parameters

- **scope_dict** (*dict*) – output of all menescope runs
- **seeds_status_dict** (*dict*) – production status of seeds in all menescope runs
- **seeds** (*str*) – SBML seeds file

Returns
union of all the individual scopes

Return type
set

metage2metabo.m2m.individual_scope.**indiv_scope_on_species**(*sbml_path*, *bname*, *seeds_path*)

Run Menetools and analyse individual metabolic capabilities on a sbml.

Parameters

- **sbml_path** (*str*) – path to SBML file
- **bname** (*str*) – name linked to SBML file
- **seeds_path** (*str*) – path to SBML seeds file

Returns
[boolean error, bname, dictionary containing menescope results]

Return type
list

metage2metabo.m2m.individual_scope.**indiv_scope_run**(*sbml_dir*, *seeds*, *output_dir*, *cpu_number=1*)

Run Menetools and analyse individual metabolic capabilities.

Parameters

- **sbml_dir** (*str*) – directory of SBML files
- **seeds** (*str*) – SBML seeds file
- **output_dir** (*str*) – directory for results
- **cpu_number** (*int*) – number of CPU to use for multiprocessing

Returns
path to output file for scope from Menetools analysis

Return type
str

metage2metabo.m2m.individual_scope.**iscope**(*sbmldir*, *seeds*, *out_dir*, *cpu_number=1*)

Compute individual scopes (reachable metabolites) for SBML files in a directory.

Parameters

- **sbmldir** (*str*) – SBML files directory
- **seeds** (*str*) – SBML seeds file
- **out_dir** (*str*) – output directory

- **cpu_number** (*int*) – number of CPU to use for multiprocessing

Returns
union of reachable metabolites for all metabolic networks

Return type
set

metage2metabo.m2m.individual_scope.**reverse_scope**(*scope_dict*, *output_dir*)

Reverse a scope dictionary by focusing on metabolite producers.

Parameters

- **scope_dict** (*dict*) – dict of scope
- **output_dir** (*str*) – path to output directory

Returns
paths to the JSON and TSV outputs

Return type
(str, str)

Community scope

metage2metabo.m2m.community_scope.**comm_scope_run**(*instance*, *output_dir*, *host_mn=None*)

Run Miscoto_scope and analyse community metabolic capabilities

Parameters

- **instance** (*str*) – instance filepath
- **output_dir** (*str*) – directory for results
- **host_mn** (*str*) – metabolic network file for host

Returns
microbiota scope dict: contribution of microbes to the scope

Return type
set

metage2metabo.m2m.community_scope.**cscope**(*sbmldir*, *seeds*, *out_dir*, *targets_file=None*, *host=None*)

Run community scope.

Parameters

- **sbmldir** (*str*) – SBML files directory
- **seeds** (*str*) – SBML file for seeds
- **out_dir** (*str*) – output directory
- **targets_file** (*str*) – targets file
- **host** (*str, optional*) – Defaults to None. Host metabolic network (SBML)

Returns
instance file (str) and community scope (set)

Return type
tuple

metage2metabo.m2m.community_scope.**instance_community**(*sbml_dir*, *seeds*, *output_dir*, *targets_file=None*, *host_mn=None*)

Create ASP instance for community analysis.

Parameters

- **sbml_dir** (*str*) – directory of symbionts SBML files
- **seeds** (*str*) – seeds SBML file
- **output_dir** (*str*) – directory for results
- **targets_file** (*str*) – targets file
- **host_mn** (*str*) – metabolic network file for host

Returns
instance filepath

Return type
str

metage2metabo.m2m.community_scope.**reverse_cscope**(*bact_contrib*, *reverse_dict*, *output_dir*)

Reverse a scope dictionary by focusing on metabolite producers.

Parameters

- **bact_contrib** (*dict*) – dict of bacteria contributions to community scope
- **reverse_dict** (*dict*) – dict of metabolite producers in community
- **output_dir** (*str*) – path to output directory

Returns
paths to the JSON and TSV outputs

Return type
(str, str)

Cooperation potential (added-value)

metage2metabo.m2m.community_addedvalue.**addedvalue**(*iscope_rm*, *cscope_rm*, *out_dir*)

Compute the added value of considering interaction with microbiota metabolism rather than individual metabolisms.

Parameters

- **iscope_rm** (*set*) – union of metabolites in all individual scopes
- **cscope_rm** (*set*) – metabolites reachable by community/microbiota
- **out_dir** (*str*) – output directory

Returns
set of metabolites that can only be reached by a community

Return type
set

Minimal community

metage2metabo.m2m.minimal_community.**compute_mincom**(*instancefile*, *miscoto_dir*)

Run minimal community selection and analysis.

Parameters

- **instancefile** (*str*) – filepath to instance file
- **miscoto_dir** (*str*) – directory with results

Returns
results of miscoto_mincom analysis

Return type
dict

metage2metabo.m2m.minimal_community.**mincom**(*instance_w_targets*, *seeds*, *targets*, *out_dir*)

Compute minimal community selection and show analyses.

Parameters

- **instance_w_targets** (*str*) – ASP instance filepath
- **seeds** (*str*) – seeds filepath
- **targets** (*str*) – targets set
- **out_dir** (*str*) – results directory

2.7.2 SBML Management

metage2metabo.sbml_management.**compare_seeds_and_targets**(*seedfile*, *targetfile*)

Returns the intersection of the seeds and the targets

Parameters

- **seedfile** (*str*) – path to seeds SBML file
- **targetfile** (*str*) – path to targets SBML file

Returns
intersection of seeds and targets

Return type
set

metage2metabo.sbml_management.**create_species_sbml**(*metabolites*, *outputfile*)

Create a SBML files with a list of species containing metabolites of the input set. Check if there are forbidden SBML characters in the metabolite IDs/ If yes, exit.

Parameters

- **metabolites** (*set*) – set of metabolites
- **outputfile** (*str*) – SBML file to be written

metage2metabo.sbml_management.**get_compounds**(*sbml_file*)

Get compound from sbml

Parameters
sbml_file (*str*) – SBML file

Returns
compound

Return type
list

metage2metabo.sbml_management.**pgdb_to_sbml**(*pgdb_dir*, *output_dir*, *noorphan_bool*, *padmet_bool*, *sbml_level*, *cpu*)

Turn Pathway Tools PGDBs into SBML2 files using Padmet

Parameters

- **pgdb_dir** (*str*) – PGDB directory
- **output_dir** (*str*) – results directory
- **noorphan_bool** (*bool*) – ignores orphan reactions if True
- **padmet_bool** (*bool*) – creates padmet files if True
- **sbml_level** (*int*) – SBML level
- **cpu** (*int*) – number of CPU for multi-process

Returns
SBML directory if successful

Return type
sbml_dir (str)

metage2metabo.sbml_management.**run_pgdb_to_sbml**(*species_multiprocess_data*)

Turn PGDBs into SBML2 using multi-processing.

Parameters
species_multiprocess_data (*list*) – pathname to species pgdb dir, pathname to species sbml file

Returns
Check if sbml file exists

Return type
sbml_check (bool)

2.7.3 Utils

metage2metabo.utils.**check_absolute_path**(*directory*, *target*)

Check if the extracted element is inside the output directory. If not, it is a potential path traversal attempt.

Parameters

- **directory** (*str*) – path to output directory for extraction.
- **target** (*str*) – path of file contained in tar file.

metage2metabo.utils.**check_program**(*program*)

Check whether Pathway Tools is in the PATH

Returns
True if Pathway Tools is in the PATH, False otherwise

Return type
bool

metage2metabo.utils.**file_or_folder**(*variable_folder_file*)

Check if the variable is file or a folder

Parameters
variable_folder_file (*str*) – path to a file or a folder

Returns
{name of input file: path to input file}

Return type
dict

metage2metabo.utils.**get_basename**(*filepath*)

Return the basename of given filepath.

Parameters
filepath (*str*) – path to a file

Returns
basename

Return type
str

```
>>> basename('~/an/interesting/file.txt')
'file
```

metage2metabo.utils.**get_extension**(*filepath*)

Get the extension of a filepath

Parameters
filepath (*str*) – path to a file

Returns
extention of the file

Return type
str

```
>>> extension('~/an/interesting/file.lp')
'lp'
>>> extension('nothing')
''
>>> extension('nothing.important')
'important'
```

metage2metabo.utils.**is_valid_dir**(*dirpath*)

Return True if directory exists or can be created (then create it)

Parameters
dirpath (*str*) – path of directory

Returns
True if dir exists, False otherwise

Return type
bool

metage2metabo.utils.**is_valid_file**(*filepath*)

Return True if filepath exists

Parameters
filepath (*str*) – path to file

Returns
True if path exists, False otherwise

Return type
bool

metage2metabo.utils.**is_valid_path**(*filepath*)

Return True if filepath is valid

Parameters
filepath (*str*) – path to file

Returns
True if path exists, False otherwise

Return type
bool

metage2metabo.utils.**safe_tar_extract_all**(*tar_file*, *outdir*)

Perform a sanitized check to ensure no file outside the output folder will be modified.

Parameters

- **tar_file** (*str*) – path to tar file.
- **outdir** (*str*) – path to output directory for extraction.

2.7.4 m2m_analysis

m2m_analysis workflow

metage2metabo.m2m_analysis.m2m_analysis_workflow.**run_analysis_workflow**(*sbml_folder*, *target_folder_file*, *seed_file*, *output_dir*, *taxon_file*, *oog_jar*, *host_file=None*, *taxonomy_level='phylum'*)

Run the whole m2m_analysis workflow

Parameters

- **sbml_folder** (*str*) – sbml directory
- **target_folder_file** (*str*) – targets file or folder containing multiple sbmls
- **seed_file** (*str*) – seeds file
- **output_dir** (*str*) – results directory
- **taxon_file** (*str*) – mpwt taxon file for species in sbml folder
- **oog_jar** (*str*) – path to OOG jar file
- **host_file** (*str*) – metabolic network file for host
- **taxonomy_level** (*str*) – taxonomy level, must be: phylum, class, order, family, genus or species.

Enumeration of Minimal communities

metage2metabo.m2m_analysis.enumeration.**convert_groups_to_equation**(*bacterial_groups*)

Convert bacterial groups (from extract_groups_from_enumeration) to boolean equation.

Parameters
bacterial_groups (*list*) – list of frozenset containing each different group of the community

Returns
string representing the boolean equation of minimal communities

Return type
boolean_equation (str)

metage2metabo.m2m_analysis.enumeration.**enumeration**(*sbml_folder*, *target_file*, *seed_file*, *output_json*, *host_file*)

Run miscoto enumeration on one target file

Parameters

- **sbml_folder** (*str*) – sbml directory
- **target_file** (*str*) – targets file
- **seed_file** (*str*) – seeds file
- **output_json** (*str*) – path to json output
- **host_file** (*str*) – metabolic network file for host

Returns
path to output json

Return type
str

metage2metabo.m2m_analysis.enumeration.**enumeration_analysis**(*sbml_folder*, *target_folder_file*, *seed_file*, *output_dir*, *host_file=None*)

Run miscoto enumeration on input data

Parameters

- **sbml_folder** (*str*) – sbml directory
- **target_folder_file** (*str*) – targets file or folder containing multiple sbmls
- **seed_file** (*str*) – seeds file
- **output_dir** (*str*) – results directory
- **host_file** (*str*) – metabolic network file for host

Returns
{target_filename_without_extension: json_output_path}

Return type
dict

metage2metabo.m2m_analysis.enumeration.**extract_groups_from_enumeration**(*results*)

From the results of the enumeration computes the boolean eaqution of the minimal communities. It is a very simple method that will fail for enumeration with numerous and complex combinations.

Parameters
results (*dict*) – results dictionary of miscoto for the enumeration.

Returns
list of frozenset containing each different group of the community

Return type
bacterial_groups (list)

Creation of graph solution

metage2metabo.m2m_analysis.solution_graph.**create_gml**(*json_paths*, *target_paths*, *output_dir*, *taxon_file=None*)

Create solution graph from miscoto output and compute stats

Parameters

- **json_paths** (*str*) – {target: path_to_corresponding_json}
- **target_paths** (*str*) – {target: path_to_corresponding_sbml}
- **output_dir** (*str*) – results directory
- **taxon_file** (*str*) – mpwt taxon file for species in sbml folder

metage2metabo.m2m_analysis.solution_graph.**graph_analysis**(*json_file_folder*, *target_folder_file*, *output_dir*, *taxon_file=None*, *taxonomy_level='phylum'*)

Run the graph creation on miscoto output

Parameters

- **json_file_folder** (*str*) – json file or folder containing multiple jsons
- **target_folder_file** (*str*) – targets file or folder containing multiple sbmls
- **output_dir** (*str*) – results directory
- **taxon_file** (*str*) – mpwt taxon file for species in sbml folder
- **taxonomy_level** (*str*) – taxonomy level, must be: phylum, class, order, family, genus or species.

Returns
path to folder containing gml results

Return type
str

Compression of graph

metage2metabo.m2m_analysis.graph_compression.**bbl_to_html**(*bbl_input*, *html_output*)

Powergraph website creation. This create a folder with html/CSS/JS files. By using the index.html file in a browser, user can see the powergraph.

Parameters

- **bbl_input** (*str*) – bbl input file
- **html_output** (*str*) – html output file

metage2metabo.m2m_analysis.graph_compression.**bbl_to_svg**(*oog_jar*, *bbl_input*, *svg_output*)

Powergraph picture creation

Parameters

- **oog_jar** (*str*) – path to oog jar file
- **bbl_input** (*str*) – bbl input file
- **svg_output** (*str*) – svg output file

metage2metabo.m2m_analysis.graph_compression.**check_oog_jar_file**(*oog_jar*)

Check Oog jar file

Parameters
oog_jar (*str*) – path to oog jar file

metage2metabo.m2m_analysis.graph_compression.**compression**(*gml_input*, *bbl_output*)

Solution graph compression

Parameters

- **gml_input** (*str*) – gml file
- **bbl_output** (*str*) – bbl output file

metage2metabo.m2m_analysis.graph_compression.**convert_taxon_id**(*taxon_id*)

Some taxon IDs are converted by powergrasp. Especially some strings are replaced by their Unicode ints. This function replaces these codes by the corresponding string.

Parameters
taxon_id (*str*) – taxon ID with potential Unicode int.

Returns
converted taxon ID

Return type
taxon_id (str)

metage2metabo.m2m_analysis.graph_compression.**merge_html_css_js**(*html_output*, *merged_html_path*)

Merge HTML/CSS/JS files into one HTML file

Parameters

- **html_output** (*str*) – path to html folder (containing css, html and css files)
- **merged_html_path** (*str*) – path to the output merged html file

metage2metabo.m2m_analysis.graph_compression.**powergraph_analysis**(*enumeration_json_folder*, *gml_input_file_folder*, *output_folder*, *oog_jar=None*, *taxon_file=None*, *taxonomy_level='phylum'*, *test_powergraph=True*)

Run the graph compression and picture creation

Parameters

- **enumeration_json_folder** (*str*) – path to the enumeration json folder or file
- **gml_input_file_folder** (*str*) – path to the gml folder or the gml file
- **output_folder** (*str*) – path to the output folder

- **oog_jar** (*str*) – path to OOG jar file
- **taxon_file** (*str*) – mpwt taxon file for species
- **taxonomy_level** (*str*) – taxonomy level, must be: phylum, class, order, family, genus or species
- **test_powergraph** (*bool*) – boolean to decide if the powergraph combinations must be tested to check for use of heuristics

metage2metabo.m2m_analysis.graph_compression.**test_powergraph_heuristics**(*enumeration_json_file*, *powergraph_bubble_file*, *output_minimal_equations_folder*, *taxon_species*)

PowerGrASP can use heuristics to compress the graph and creates powergraph representation. So some powergraphs visualisation are not correct according to the combination of the enumeration of minimal solution. This function tests the powergraph to see if the viusalized combinations corresponds to the one of the enumeration. If no heuristics have been used, then the function tries to create a boolean equation summarizing the powergraph.

Parameters

- **enumeration_json_file** (*str*) – path to enumeration json file
- **powergraph_bubble_file** (*str*) – path to the bbl file containing powergraph
- **output_minimal_equations_folder** (*str*) – output fodler for minimal boolean equation of powernodes
- **taxon_species** (*dict*) – associate organism ID as key with taxon name as value

metage2metabo.m2m_analysis.graph_compression.**update_js**(*html_output*, *essentials*, *alternatives*)

Update graph.js to add colors for essential and alternative symbionts.

Parameters

- **html_output** (*str*) – path to html folder (containing js subfolder with gaph.js)
- **essentials** (*list*) – list of essential symbionts
- **alternatives** (*list*) – list of alternative symbionts

metage2metabo.m2m_analysis.graph_compression.**update_js_taxonomy**(*html_output*, *taxon_colors*, *essentials*, *alternatives*)

Update graph.js to add colors according to taxon.

Parameters

- **html_output** (*str*) – path to html folder (containing js subfolder with gaph.js)
- **taxon_colors** (*dict*) – dictionary {taxon_name: associated_color}
- **essentials** (*list*) – list of essential symbionts
- **alternatives** (*list*) – list of alternative symbionts

metage2metabo.m2m_analysis.graph_compression.**update_svg**(*svg_file*, *essentials*, *alternatives*)

Update svg file to add colors for essential and alternative symbionts.

Parameters

- **svg_file** (*str*) – path to svg file
- **essentials** (*list*) – list of essential symbionts

- **alternatives** (*list*) – list of alternative symbionts

metage2metabo.m2m_analysis.graph_compression.**update_svg_taxonomy**(*svg_file*, *taxon_colors*)

Update svg file to add colors for each taxon

Parameters

- **svg_file** (*str*) – path to svg file
- **taxon_colors** (*dict*) – dictionary {taxon_name: associated_color}

taxonomy

metage2metabo.m2m_analysis.taxonomy.**extract_taxa**(*mpwt_taxon_file*, *taxon_output_file*, *tree_output_file*, *taxonomy_level='phylum'*)

From NCBI taxon ID, extract taxonomy rank and create a tree file

Parameters

- **mpwt_taxon_file** (*str*) – mpwt taxon file for species in sbml folder
- **taxon_output_file** (*str*) – path to taxonomy output file
- **tree_output_file** (*str*) – path to tree output file
- **taxonomy_level** (*str*) – taxonomy level, must be: phylum, class, order, family, genus or species.

metage2metabo.m2m_analysis.taxonomy.**get_taxon**(*taxonomy_file_path*)

From the taxonomy file (created by extract_taxa) create a dictionary and a list linking taxon and species

Parameters
taxonomy_file_path (*str*) – path to the taxonomy_file

Returns
associate organism ID as key with taxon name as value all_taxa (list): list all taxa in file

Return type
taxon_named_species (dict)

2.7.5 Main

metage2metabo.__main__.**create_metadata**(*dict_args*, *duration*, *metadata_json_file*)

Create metadata from args and package versions.

Parameters

- **dict_args** (*dict*) – dict of args given to argparse
- **duration** (*int*) – time of the run
- **metadata_json_file** (*str*) – pat hto metadata output file

metage2metabo.__main__.**main**()

Run programm.

metage2metabo.__main__.**main_added_value**(*sbmldir*, *seeds*, *outdir*, *host*)

Run addedvalue command.

Parameters

- **sbmldir** (*str*) – SBML file directory
- **seeds** (*str*) – SBML file for seeds
- **outdir** (*str*) – results directory
- **host** (*str*) – SBML file for host

metage2metabo.__main__.**main_cscope**(**allargs*)

Run cscope command.

metage2metabo.__main__.**main_iscope**(**allargs*)

Run iscope command.

metage2metabo.__main__.**main_metacom**(**allargs*)

Run main workflow.

metage2metabo.__main__.**main_mincom**(*sbmldir*, *seedsfiles*, *outdir*, *targets*, *host*)

Run mincom command.

Parameters

- **sbmldir** (*str*) – SBML files directory
- **seedsfiles** (*str*) – SBML file for seeds
- **outdir** (*str*) – results directory
- **targets** (*str*) – targets SBML file
- **host** (*str*) – SBML file for host

metage2metabo.__main__.**main_recon**(**allargs*)

Run recon command.

metage2metabo.__main__.**main_seeds**(*metabolites_file*, *outdir*)

Run seeds command.

Parameters

- **metabolites_file** (*str*) – text file with metabolites IDs, one per line
- **outdir** (*str*) – Results directory

metage2metabo.__main__.**main_test**(*outdir*, *cpu*)

Run test command.

Parameters

- **outdir** (*str*) – directory containing the test data and the test output
- **cpu** (*int*) – number of cpu to use (recommended: 2)

metage2metabo.__main__.**main_workflow**(**allargs*)

Run main workflow.

2.8 m2m_analysis

m2m_analysis is a supplementary command installed by metage2metabo. It is separated from m2m because it has heavy dependencies. Also, m2m_analysis steps can be time consuming.

2.8.1 Requirements

m2m_analysis needs:

- Oog Power Graph Command line tool: used to create a svg file of the power graph. It is a jar file (`Oog.jar` compiled for Java 6), so you need at least Java 6.
- These python packages:
  - networkx: to create graph from miscoto results
  - ete3: to add taxonomy information on the graph if you used mpwt taxon file
  - powergrasp: to compress networkx graph (which required graphviz)

2.8.2 Presentation

m2m_analysis goes deeper in the analysis compare to m2m. In m2m_analysis, the enumeration of all solution is computed, this step is far more time consuming than the others (union or intersection).

This first step is done by `m2m_analysis enum`. It will enumerate all the possible solutions (minimal communities), select the optimal ones and then create the json file containing all the optimal solutions.

In a second step (`m2m_analysis graph`), the optimal solutions from the enumeration are stored in a graph. The nodes of this graph are each organism present in at least one solution. An edge connects two nodes, only if the two organisms (represented by the node) co-occur in at least one of the enumerated communities.

The last step (`m2m_analysis powergraph`) compresses the graph into a power graph (in bbl format). Then it creates a svg picture of this power graph.

2.8.3 m2m Tutorial

Test data is avaible in the Github repository. It contains enough data to run the different subcommands.

m2m_analysis enum

`m2m_analysis enum` runs miscoto with enumeration on a set of target.

It uses the following mandatory inputs (run `m2m_analysis enum --help` for optional arguments):

-n directory	directory of metabolic networks, in SBML format
-s file	seeds SBML file
-t directory	targets SBML file or folder containing multiple targets SBML files
-o directory	output directory for results

Optional arguments:

-q	quiet mode
-m file	host metabolic network in SBML

```
m2m_analysis enum -n toy_bact -s metabolic_data/seeds_toy.sbml -t metabolic_data/targets_
↪toy.sbml -o output_directory
```

- **standard output**

```
##############################################
#                                            #
#      Enumeration of minimal communities    #
#                                            #
##############################################

######### Enumeration of solution for: targets_toy #########
/shared/Softwares/git/miscoto/miscoto/encodings/community_soup.lp
######### Enumeration of minimal communities #########
5 minimal communities (each containing 13 species) producing the target␣
↪metabolites
######### Key species: Union of minimal communities #########
# Bacteria occurring in at least one minimal community enabling the␣
↪producibility of the target metabolites given as inputs
Key species = 17
GCA_003437905
GCA_003437255
GCA_003437345
GCA_003437785
GCA_003437595
GCA_003437715
GCA_003437375
GCA_003437325
GCA_003437815
GCA_003437295
GCA_003437175
GCA_003437885
GCA_003437945
GCA_003437665
GCA_003437195
GCA_003437055
GCA_003438055
######### Essential symbionts: Intersection of minimal communities #########
# Bacteria occurring in ALL minimal community enabling the producibility of␣
↪the target metabolites given as inputs
Essential symbionts = 12
GCA_003437815
GCA_003437295
GCA_003437905
GCA_003437255
GCA_003437885
GCA_003437665
GCA_003437195
GCA_003437595
GCA_003437055
GCA_003438055
GCA_003437715
GCA_003437375
```

(continues on next page)

(continued from previous page)

```
######### Alternative symbionts: Difference between Union and Intersection ####
↪#####
# Bacteria occurring in at least one minimal community but not all minimal␣
↪community enabling the producibility of the target metabolites given as␣
↪inputs
Alternative symbionts = 5
GCA_003437325
GCA_003437945
GCA_003437175
GCA_003437785
GCA_003437345
--- Enumeration runtime 4.64 seconds ---

--- Total runtime 4.65 seconds ---
```

- **files output**

```
output_directory
├── json
│   ├── targets_toy.json
├── m2m_analysis_enum.log
├── m2m_analysis_metadata.json
```

m2m_analysis graph

`m2m_analysis graph` creates the graph containing the solutions.

It uses the following mandatory inputs (run `m2m_analysis graph --help` for optional arguments):

-j directory	directory of miscoto output JSONs or single JSON
-t directory	targets SBML file or folder containing multiple targets SBML files
-o directory	output directory for results

Optional arguments:

-q	quiet mode
--taxon file	mpwt taxon file
--level LEVEL	Taxonomy level, must be: phylum, class, order, family, genus or species. By default, it is phylum.

You can use the taxon file from gut experience.

```
m2m_analysis graph -j output_directory/json -t metabolic_data/targets_toy.sbml -o output_
↪directory
```

- **standard output**

```
################################################
#                                              #
#         Solution graph creation              #
#                                              #
################################################
```

(continues on next page)

(continued from previous page)

```
######### Graph of targets_toy #########
Number of nodes: 17
Number of edges: 126
--- Graph runtime 0.01 seconds ---

--- Total runtime 0.01 seconds ---
```

- **files output**

```
output_directory
├── gml
│   ├── targets_toy.gml
├── key_species.json
├── key_species_stats.tsv
├── m2m_analysis_graph.log
├── miscoto_stats.txt
├── m2m_analysis_metadata.json
```

m2m_analysis powergraph

`m2m_analysis powergraph` compresses the graph and create a svg picture.

It uses the following mandatory inputs (run `m2m_analysis powergraph --help` for optional arguments):

-j JSON_DIR_OR_FILE Folder containing JSON files of single JSON file containing miscoto enumeration results

-g file directory of GML files or a GML file

-o directory output directory for results

Optional arguments:

-q quiet mode

--oog file Oog jar file (present in external_dependencies folder of the github repository)

--taxon TAXON Mpwt taxon file

--level LEVEL Taxonomy level, must be: phylum, class, order, family, genus or species. By default, it is phylum.

```
m2m_analysis powergraph --oog Oog.jar -g output_directory/gml -j output_directory/json -
↪o output_directory
```

- **standard output**

```
################################################
#                                              #
#  Compression and visualisation of graph      #
#                                              #
################################################

######### Graph compression: targets_toy #########
Number of powernodes: 3
```

(continues on next page)

(continued from previous page)

```
Number of poweredges: 2
Compression runtime 1.52 seconds ---

######### Test powergraph heuristics: targets_toy #########
Same combinations between theorical (5) and solution (5)
The powergraph seems to be an optimal representation of the solutions.

Same combinations between theorical (5) and the enumeration of solutions by␣
↪m2m_analysis  (5)
The powergraph seems to be an optimal representation of the solutions.

Same number of solution between the computed combinations from powernodes (5)␣
↪and the enumeration of solutions by m2m_analysis (5)
But this does not indicate that the powernodes are an optimal representation␣
↪but that they contain the solution.

It seems that there are no heuristics in powergraph so it could be possible to␣
↪create a boolean equation.
######### Boolean equation of minimal communities #########
The boolean equation can only be created for simple case (without too many␣
↪combinations).
Boolean equation seems good, as it has the same combinations (5) than the one␣
↪from enumeration (5).
Boolean equation:
(( GCA_003437905 ) &
( GCA_003437255 ) &
( GCA_003437785 | GCA_003437345 | GCA_003437175 | GCA_003437325 | GCA_
↪003437945 ) &
( GCA_003437595 ) &
( GCA_003437715 ) &
( GCA_003437375 ) &
( GCA_003437815 ) &
( GCA_003437295 ) &
( GCA_003437885 ) &
( GCA_003437665 ) &
( GCA_003437195 ) &
( GCA_003437055 ) &
( GCA_003438055 ) )
######### PowerGraph visualization: targets_toy #########
Creation of the powergraph website accessible at output_directory/html/targets_
↪toy
Creation of the powergraph svg accessible at output_directory/svg
*******************************************************************************
*  Oog - PowerGraph Library (Matthias Reimann, c 2006-2012)                   ␣
↪*
*  PowerGraph Analysis through the command line interface of Oog              ␣
↪*
*                                                                             ␣
↪*
*  Please cite us: Royer L, Reimann M, Andreopoulos B, Schroeder M            ␣
↪*
*  (2008) Unraveling Protein Networks with Power Graph Analysis.              ␣
```

(continues on next page)

(continued from previous page)

```
↪*
*  PLoS Comput Biol 4(7): e1000108                                              ␣
↪*
*                                                                               ␣
↪*
*  Contact: reimann@biotec.tu-dresden.de                                        ␣
↪*
********************************************************************************
<II> Current time: 2024/03/01 11:56:46
<II> Oog build: Oog_build_2012.04.17_14.16.48

<II> Working directory: . (/shared/Softwares/git/metage2metabo/test/metabolic_
↪data/)
<II> Graph file directories: [.]
<II> Output directory: output_directory/svg
<II> Loading graph (targets_toy.bbl) ... 21ms
<II> Arrange Graph ... Exception in thread "PowerGraphArranger" java.lang.
↪IndexOutOfBoundsException: Index 20 out of bounds for length 20
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.
↪java:64)
    at java.base/jdk.internal.util.Preconditions.
↪outOfBoundsCheckIndex(Preconditions.java:70)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.
↪java:248)
    at java.base/java.util.Objects.checkIndex(Objects.java:374)
    at java.base/java.util.ArrayList.get(ArrayList.java:459)
    at org.mattlab.eaglevista.graph.OogGraph.getID_(OogGraph.java:2703)
    at org.mattlab.eaglevista.graph.OogPGArranger.arrangeRec(OogPGArranger.
↪java:361)
    at org.mattlab.eaglevista.graph.OogPGArranger.arrange(OogPGArranger.
↪java:327)
    at org.mattlab.eaglevista.graph.OogPGArranger.run(OogPGArranger.java:271)
    at java.base/java.lang.Thread.run(Thread.java:829)
4001ms (14ms)
<II> Create SVG ... 275ms
<II> Image written (output_directory/svg/targets_toy.bbl.svg)
--- Powergraph runtime 6.06 seconds ---

--- Total runtime 6.08 seconds ---
```

- **files output**

```
output_directory
├── bbl
│   ├── targets_toy.bbl
├── html
│   ├── targets_toy
│   │   ├── js
│   │   │   ├── cytoscape.min.js
│   │   │   ├── cytoscape-cose-bilkent.js
│   │   │   ├── graph.js
│   │   ├── index.html
```

(continues on next page)

(continued from previous page)

```
│   │   ├── style.css
│   ├── targets_toy_powergraph.html
├── svg
│   ├── targets_toy.bbl.svg
├── m2m_analysis_powergraph.log
├── m2m_analysis_metadata.json
```

This command creates the following svg (node colords: dark pink for essential symbionts and blue for alternative symbionts):

m2m_analysis workflow

`m2m_analysis workflow` runs the all m2m_analysis workflow.

It uses the following mandatory inputs (run `m2m_analysis workflow --help` for optional arguments):

-n directory	directory of metabolic networks, in SBML format
-s file	seeds SBML file
-t directory	targets SBML file or folder containing multiple targets SBML files
-o directory	output directory for results

Optional arguments:

-q	quiet mode
-m file	host metabolic network in SBML
--taxon file	mpwt taxon file
--oog file	Oog jar file
--level LEVEL	Taxonomy level, must be: phylum, class, order, family, genus or species. By default, it is phylum.

```
m2m_analysis workflow -n toy_bact -s metabolic_data/seeds_toy.sbml -t metabolic_data/
↪targets_toy.sbml -o output_directory --oog Oog.jar --taxon taxon_id.tsv
```

- **standard output**

```
################################################
#                                              #
#      Enumeration of minimal communities      #
#                                              #
################################################

######### Enumeration of solution for: targets_toy #########
/shared/Softwares/git/miscoto/miscoto/encodings/community_soup.lp
######### Enumeration of minimal communities #########
5 minimal communities (each containing 13 species) producing the target␣
↪metabolites
######### Key species: Union of minimal communities #########
# Bacteria occurring in at least one minimal community enabling the␣
↪producibility of the target metabolites given as inputs
Key species = 17
```

(continues on next page)

(continued from previous page)

```
GCA_003437195
GCA_003437885
GCA_003437055
GCA_003437815
GCA_003437375
GCA_003437665
GCA_003437175
GCA_003437945
GCA_003437715
GCA_003438055
GCA_003437345
GCA_003437905
GCA_003437325
GCA_003437255
GCA_003437595
GCA_003437785
GCA_003437295
######### Essential symbionts: Intersection of minimal communities #########
# Bacteria occurring in ALL minimal community enabling the producibility of␣
↪the target metabolites given as inputs
Essential symbionts = 12
GCA_003437195
GCA_003437885
GCA_003437055
GCA_003437815
GCA_003437375
GCA_003437905
GCA_003437665
GCA_003437255
GCA_003437595
GCA_003437715
GCA_003437295
GCA_003438055
######### Alternative symbionts: Difference between Union and Intersection ####
↪#####
# Bacteria occurring in at least one minimal community but not all minimal␣
↪community enabling the producibility of the target metabolites given as␣
↪inputs
Alternative symbionts = 5
GCA_003437175
GCA_003437785
GCA_003437345
GCA_003437325
GCA_003437945
--- Enumeration runtime 4.73 seconds ---


################################################
#                                              #
#         Solution graph creation              #
#                                              #
################################################
```

(continues on next page)

(continued from previous page)

```
######### Extract taxon information from taxon_id.tsv. #########
No taxon_id for GCA_003437378 in file taxon_id.tsv.
--- Taxonomy runtime 0.02 seconds ---

######### Graph of targets_toy #########
Number of nodes: 17
Number of edges: 126
--- Graph runtime 0.03 seconds ---


################################################
#                                              #
#  Compression and visualisation of graph      #
#                                              #
################################################

######### Graph compression: targets_toy #########
Number of powernodes: 3
Number of poweredges: 2
Compression runtime 1.55 seconds ---

######### Test powergraph heuristics: targets_toy #########
Same combinations between theorical (5) and solution (5)
The powergraph seems to be an optimal representation of the solutions.

Same combinations between theorical (5) and the enumeration of solutions by␣
↪m2m_analysis  (5)
The powergraph seems to be an optimal representation of the solutions.

Same number of solution between the computed combinations from powernodes (5)␣
↪and the enumeration of solutions by m2m_analysis (5)
But this does not indicate that the powernodes are an optimal representation␣
↪but that they contain the solution.

It seems that there are no heuristics in powergraph so it could be possible to␣
↪create a boolean equation.
######### Boolean equation of minimal communities #########
The boolean equation can only be created for simple case (without too many␣
↪combinations).
Boolean equation seems good, as it has the same combinations (5) than the one␣
↪from enumeration (5).
Boolean equation:
(( GCA_003437195 ) &
( GCA_003437885 ) &
( GCA_003437055 ) &
( GCA_003437815 ) &
( GCA_003437375 ) &
( GCA_003437665 ) &
( GCA_003437175 | GCA_003437785 | GCA_003437345 | GCA_003437325 | GCA_
↪003437945 ) &
( GCA_003437715 ) &
```

(continues on next page)

(continued from previous page)

```
( GCA_003438055 ) &
( GCA_003437905 ) &
( GCA_003437255 ) &
( GCA_003437595 ) &
( GCA_003437295 ) )


Boolean equation with taxonomic name:
(( Bacillota__2 ) &
( Bacteroidota__2 ) &
( Bacillota__1 ) &
( Bacteroidota__1 ) &
( Actinomycetota__4 ) &
( Bacillota__4 ) &
( Actinomycetota__1 | Actinomycetota__6 | Actinomycetota__3 | Actinomycetota__
↪2 | Actinomycetota__7 ) &
( Actinomycetota__5 ) &
( Bacillota__6 ) &
( Bacillota__5 ) &
( Pseudomonadota__1 ) &
( Bacillota__3 ) &
( Pseudomonadota__2 ) )
######### PowerGraph visualization: targets_toy #########
Creation of the powergraph website accessible at output_directory/html/targets_
↪toy
Creation of the powergraph svg accessible at output_directory/svg
*******************************************************************************
*  Oog - PowerGraph Library (Matthias Reimann, c 2006-2012)                   ␣
↪*
*  PowerGraph Analysis through the command line interface of Oog              ␣
↪*
*                                                                             ␣
↪*
*  Please cite us: Royer L, Reimann M, Andreopoulos B, Schroeder M            ␣
↪*
*  (2008) Unraveling Protein Networks with Power Graph Analysis.              ␣
↪*
*  PLoS Comput Biol 4(7): e1000108                                            ␣
↪*
*                                                                             ␣
↪*
*  Contact: reimann@biotec.tu-dresden.de                                      ␣
↪*
*******************************************************************************
<II> Current time: 2024/03/01 11:58:24
<II> Oog build: Oog_build_2012.04.17_14.16.48

<II> Working directory: . (/shared/Softwares/git/metage2metabo/test/metabolic_
↪data/)
<II> Graph file directories: [.]
<II> Output directory: output_directory/svg
<II> Loading graph (targets_toy.bbl) ... 18ms
```

(continues on next page)

(continued from previous page)

```
<II> Arrange Graph ... Exception in thread "PowerGraphArranger" java.lang.
↪IndexOutOfBoundsException: Index 20 out of bounds for length 20
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.
↪java:64)
    at java.base/jdk.internal.util.Preconditions.
↪outOfBoundsCheckIndex(Preconditions.java:70)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.
↪java:248)
    at java.base/java.util.Objects.checkIndex(Objects.java:374)
    at java.base/java.util.ArrayList.get(ArrayList.java:459)
    at org.mattlab.eaglevista.graph.OogGraph.getID_(OogGraph.java:2703)
    at org.mattlab.eaglevista.graph.OogPGArranger.arrangeRec(OogPGArranger.
↪java:361)
    at org.mattlab.eaglevista.graph.OogPGArranger.arrange(OogPGArranger.
↪java:327)
    at org.mattlab.eaglevista.graph.OogPGArranger.run(OogPGArranger.java:271)
    at java.base/java.lang.Thread.run(Thread.java:829)
4000ms (13ms)
<II> Create SVG ... 222ms
<II> Image written (output_directory/svg/targets_toy.bbl.svg)
--- Powergraph runtime 6.01 seconds ---

--- m2m_analysis runtime 10.77 seconds ---

--- Total runtime 10.79 seconds ---
```

- **files output**

```
output_directory
├── bbl
│   ├── targets_toy.bbl
├── json
│   ├── targets_toy.json
├── gml
│   ├── targets_toy.gml
├── html
│   ├── targets_toy
│   │   ├── js
│   │   │   ├── cytoscape.min.js
│   │   │   ├── cytoscape-cose-bilkent.js
│   │   │   ├── graph.js
│   │   ├── index.html
│   │   ├── style.css
│   ├── targets_toy_taxon
│   │   ├── js
│   │   │   ├── cytoscape.min.js
│   │   │   ├── cytoscape-cose-bilkent.js
│   │   │   ├── graph.js
│   │   ├── index.html
│   │   ├── style.css
│   ├── targets_toy_powergraph.html
│   ├── targets_toy_powergraph_taxon.html
```

(continues on next page)

(continued from previous page)

```
├── minimal_equation
│   ├── targets_toy
│       ├── boolean_equation.json
│       ├── minimal_equations.tsv
│       ├── powernodes_composition.json
├── svg
│   ├── targets_toy.bbl.svg
│   ├── targets_toy_taxon.bbl.svg
├── key_species.json
├── key_species_stats.tsv
├── m2m_analysis_workflow.log
├── miscoto_stats.txt
├── taxon_tree.txt
├── taxonomy_species.tsv
├── m2m_analysis_metadata.json
```

This command creates the previous svg and a new svg with the nodes colored according to their taxons:

2.8.4 m2m_analysis output files

```
output_directory
├── bbl
│   ├── targets_toy.bbl
├── json
│   ├── targets_toy.json
├── gml
│   ├── targets_toy.gml
├── html
│   ├── targets_toy
│   │   ├── js
│   │   │   ├── cytoscape.min.js
│   │   │   ├── cytoscape-cose-bilkent.js
│   │   │   ├── graph.js
│   │   ├── index.html
│   │   ├── style.css
│   ├── targets_toy_taxon (with the ``--taxon`` option)
│   │   ├── js
│   │   │   ├── cytoscape.min.js
│   │   │   ├── cytoscape-cose-bilkent.js
│   │   │   ├── graph.js
│   │   ├── index.html
│   │   ├── style.css
│   ├── targets_toy_powergraph.html
│   ├── targets_toy_powergraph_taxon.html (with the ``--taxon`` option)
├── svg
│   ├── targets_toy.bbl.svg
│   ├── targets_toy_taxon.bbl.svg (with the ``--taxon`` option)
├── minimal_equation
│   ├── targets_toy
│       ├── boolean_equation.json
```

(continues on next page)

(continued from previous page)

```
│       ├── minimal_equations.tsv
│       ├── powernodes_composition.json
├── key_species.json
├── key_species_stats.tsv
├── m2m_analysis_*.log
├── miscoto_stats.txt
├── taxon_tree.txt (with the ``--taxon`` option)
├── taxonomy_species.tsv (with the ``--taxon`` option)
├── m2m_analysis_metadata.json
```

Miscoto results

`miscoto_stats.txt`: for each target, this file summarizes the result of miscoto with: the number of targets, the size of the minimal solutions, the size of the union of the minimal solution (the key species), the size of the intersection of the minimal solutions (the essential symbionts)) and the size of the enumeration (the number of minimal solutions).

`json/*.json`: for each target, there will be a miscoto json result file. If you have given one target file, the result file will be named according to the name of the target file.

The json contains 21 keys:

- `bacteria`: organisms in the optimal solution.
- `still_unprod`: compounds unproducible by the community.
- `newly_prod`: compounds newly producible by the community.
- `producible`: compounds producible by the community.
- `union_bacteria`: organisms from all the minimal communities.
- `inter_bacteria`: organisms from the intersection of all the minimal communities.
- `one_model`: results of the optimal solution.
- `exchanged`, `union_exchanged`, `inter_exchanged` and `enum_exchanged`: the exchanged compounds by the community, this step needs a lot of resources so it is not used in m2m. If you want to use it, use miscoto with the `minexch` option.
- `key_species`: organisms from all the minimal communities.
- `essential_symbionts`: organisms in the intersection of all the minimal communities. They are occuring in all minimal solution.
- `alternative_symbionts`: organisms appearing in at least one minimal community but not in all.
- `score_optimum_inter`: the optimum score found for the intersection, it corresponds to the number of organism in the minimal community.
- `score_optimum_union`: the optimum score found for the union, it corresponds to the number of organism in the minimal community.
- `inter_targetsproducers`: the organism that have the final reaction to produce the target in the intersection. It is a dictionary, with each target as key and the organism producing these targets as value.
- `union_targetsproducers`: the organism that have the final reaction to produce the target in the union. It is a dictionary, with each target as key and the organism producing these targets as value.
- `one_model_targetsproducers`: the organism that have the final reaction to produce the target in the optimal solution. It is a dictionary, with each target as key and the organism producing these targets as value.

- `enum_bacteria`: all the minimal solutions. It is a dictionary, with a number (linked to a minimal solution) as key and the organisms in the corresponding minimal solution as value.
- `enum_targetsproducers`: the organism that have the final reaction to produce the target in the union. It is a dictionary, with a number (corresponding to a minimal solution) as key and a dictionary as value This sub-dictionary contains each target as key and the organism producing these targets as value.

Key species files

`key_species_stats.tsv`: for each target, this file contains the number of key species, essential and alternative symbionts. If you have used the "–taxon" option, the column corresponds to each taxonomic groups.

`key_species.json`: for each target, this json file contains the name of species in the key species, essential or alternative symbionts groups.

For each target, the json contains 2 keys:

- `essential_symbionts`: a dictionary with a key `data` and the name of theorganisms in the essential symbionts. If you use the "–taxon" option, the dictionary keys will be the name of taxon and the values will be the essential symbionts of this taxon.
- `alternative_symbionts`: a dictionary with a key `data` and the name of theorganisms in the alternative symbionts. If you use the "–taxon" option, the dictionary keys will be the name of taxon and the values will be the alternative symbionts of this taxon.

Solution graph

The solution graph is stored in a gml file (`gml/*.gml`). The nodes of the graph are species occuring in minimal solutions. An edge is created between two nodes when the two nodes are in the same minimal solutions.

Compressed solution graph (power graph)

The previous solution graph is then compressed into a power graph using PowerGrASP. The result file is bbl file (`bbl/*.bbl`).

You can use this file to visualize the power graph with Cytoscape. To do this you need the version 2.8.2 of Cytoscape and the CyOog plugin. When you have installed Cytoscape, put the file `CyOog.jar` in `path/to/cytoscape/install/dir/plugins/`.

m2m_analysis can also create visualization of the power graph.

Boolean equation

If the power graph is simple enough, Metage2Metabo will try to create a boolean equation summarizing the power graph.

Boolean equation is written as follow:

```
(( GCA_003437195 ) &
( GCA_003437885 ) &
( GCA_003437055 ) &
( GCA_003437815 ) &
( GCA_003437375 ) &
( GCA_003437665 ) &
( GCA_003437175 | GCA_003437785 | GCA_003437345 | GCA_003437325 | GCA_003437945 ) &
```

(continues on next page)

(continued from previous page)

```
( GCA_003437715 ) &
( GCA_003438055 ) &
( GCA_003437905 ) &
( GCA_003437255 ) &
( GCA_003437595 ) &
( GCA_003437295 ) )
```

& is for the AND operator indicating that the two groups around this operator are needed. `( GCA_003437195 ) & ( GCA_003437885 )` reads as we need both GCA_003437195 AND GCA_003437885.

| is for the OR operator indicating indicating that the the two groups around this operator are redundant. `( GCA_003437175 | GCA_003437785 | GCA_003437345 | GCA_003437325 | GCA_003437945 )` reads as we need one organism among GCA_003437175, GCA_003437785, GCA_003437345, GCA_003437325, GCA_003437945.

The results will be stored in three files:

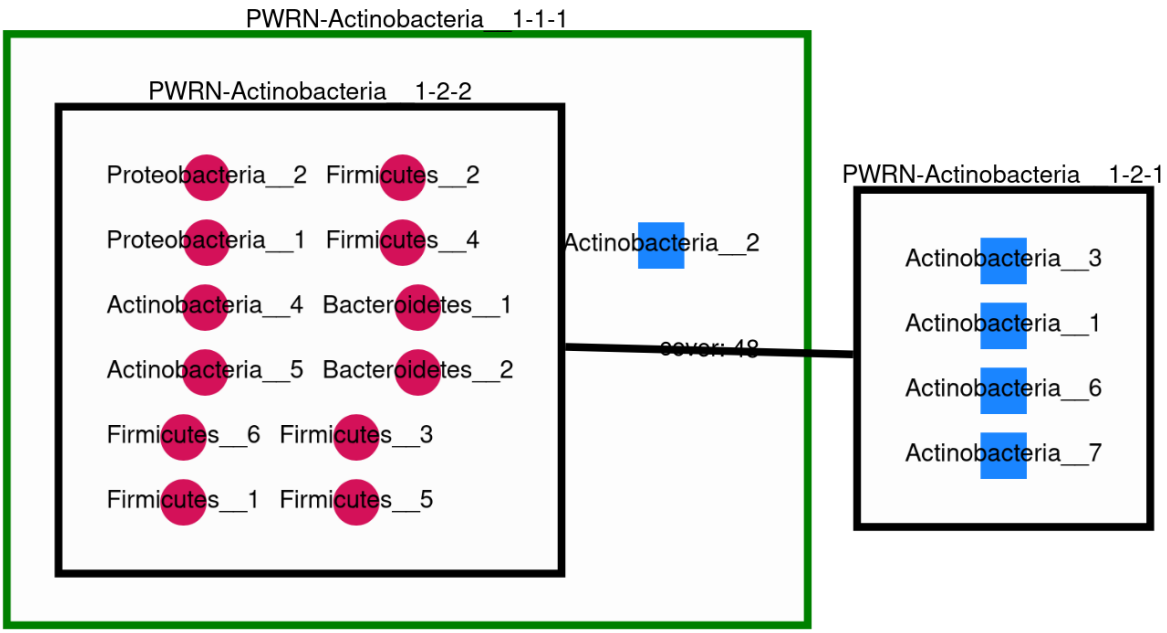
- `boolean_equation.json`: a json containing several keys: (1) `boolean_equation` the boolean equation, (2) `bacterial_groups` the identified groups associated with the boolean equation, (3) `boolean_equation_taxon` if the "–taxon" option was used the boolean equation with taxon name and (4) if the "–taxon" option was used the identified groups (with taxon name) associated with the boolean equation.
- `minimal_equations.tsv`: a tsv file containing multiple rows. Each row corresponds to a minimal solution according to the powernodes identified by powergrasp.
- `powernodes_composition.json`: a json file containing the powernode composition contained in `minimal_equations.tsv`.

Mini website

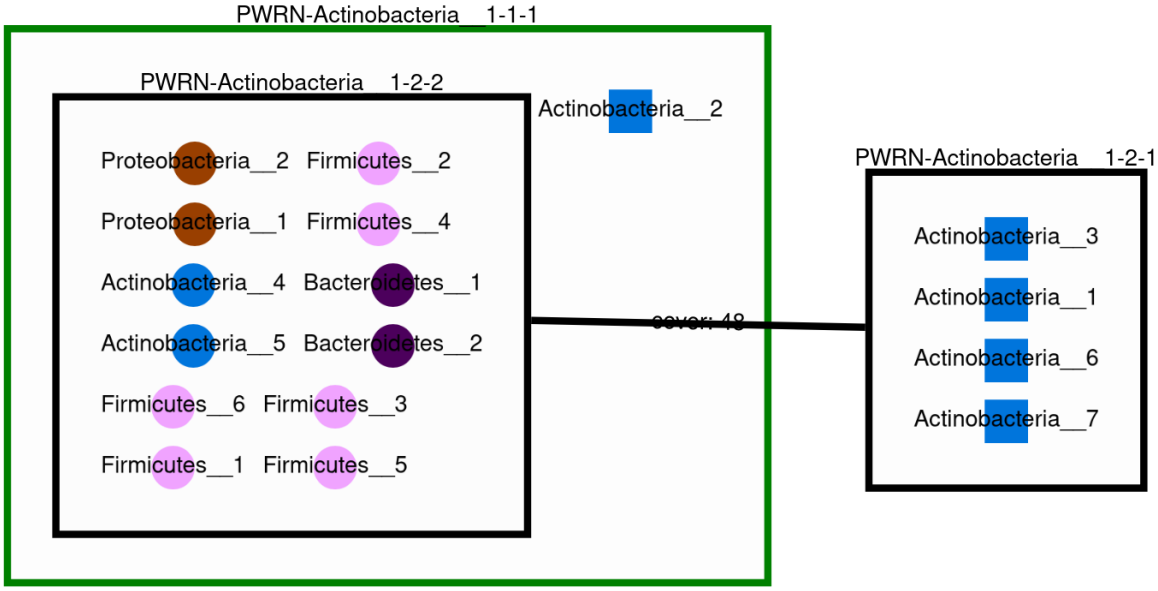
A website like interface is created using the package bubble-tools.

For each target, there will a folder in `html` folder and a html file (named after the target). The folder is created by bubble-tools and contains html, css and js files with the information for the visualization. The html file is a merged of all the files from the folder, to ease the visualization.

To view the html file, open it with a web browser (like `Firefox`). There will be nodes in two shapes (circle for essential symbionts and rectangle for alternative symbionts) and colors (dark pink circle for essential symbionts and blue rectangle for alternative symbionts). And powernode will be rectangle.



If you used the "–taxon" option, a new html file is created and labelled as `*_taxon.html` (and a corresponding folder is created). In this html file, the node will be colored according to their taxon and the shape indicated if it is an essential symbiont (circle) or an alternative symbiont (rectangle).



With the html file it is possible to interact and move the node by clicking on them.

Picture of the power graph

The compresed graph is then drawn using the Oog Power Graph Command line tool. A svg file is then created (`svg/*.svg`). The nodes will be colored in dark pink for essential symbionts and blue for alternative symbionts.

If you used the "–taxon" option, a new html file is created and labelled as `svg/*_taxon.svg`. In this file, the node will be colored according to their taxon.

Taxonomy linked files

If you have used the `--taxon`, two new files have been created:

`taxonomy_species.tsv`: it is a tsv file with 9 columns. The row corresponds to the species in your community. For each species, you will have its name in your dataset, its taxID (from taxon_id.tsv), an attributed taxonomic name (used in the power graph), then the taxonomic classification: phylum, class, order, family, genus and species.

`taxon_tree.txt`: the topology of the taxonomic classification of your species according to the NCBI taxonomy.

2.8.5 More information

Take a look at the complete tutorial in the Github repository

2.9 Troubleshooting

Several potential fix for issues we encounter.

2.9.1 *m2m_analysis*: "Can't connect to X11 window server using..."

Issue encountered when using *m2m_analysis* on a HPC:

```
<II> Create SVG ... Exception in thread "main" java.awt.AWTError: Can't connect to X11␣
↪window server using ':0' as the value of the DISPLAY variable.
    at sun.awt.X11GraphicsEnvironment.initDisplay(Native Method)
    at sun.awt.X11GraphicsEnvironment.access$200(X11GraphicsEnvironment.java:65)
    at sun.awt.X11GraphicsEnvironment$1.run(X11GraphicsEnvironment.java:115)
    at java.security.AccessController.doPrivileged(Native Method)
    at sun.awt.X11GraphicsEnvironment.<clinit>(X11GraphicsEnvironment.java:74)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:264)
    at java.awt.GraphicsEnvironment.createGE(GraphicsEnvironment.java:103)
    at java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment(GraphicsEnvironment.
↪java:82)
    at java.awt.image.BufferedImage.createGraphics(BufferedImage.java:1181)
    at org.apache.batik.svggen.SVGGraphics2D.<init>(Unknown Source)
    at org.apache.batik.svggen.SVGGraphics2D.<init>(Unknown Source)
    at org.mattlab.eaglevista.ui.GraphImageFactory.
↪getSVGGraphicsOfGraph(GraphImageFactory.java:157)
    at org.mattlab.eaglevista.ui.GraphImageFactory.createImage(GraphImageFactory.java:80)
    at org.mattlab.eaglevista.ui.EvCommandLineInterpreter.
↪runImpl(EvCommandLineInterpreter.java:466)
```

(continues on next page)

(continued from previous page)

```
    at org.mattlab.eaglevista.ui.EvCommandLineInterpreter.main(EvCommandLineInterpreter.
↪java:81)
```

One potential fix is to unset DISPLAY before launching m2m:

```
unset DISPLAY
m2m_analysys workflow...
```

2.9.2 UnicodeDecodeError when running m2m_analysis with Singularity

Error encountered when using *m2m_analysis* in a Singularity:

```
######### Creation of the powergraph website accessible at results_m2m_analysis_scfa/
↪html/metabolic_targets_scfa #########
Traceback (most recent call last):
File "/usr/local/bin/m2m_analysis", line 8, in <module>
    sys.exit(main())
File "/usr/local/lib/python3.6/dist-packages/metage2metabo/__main_analysis__.py", line␣
↪289, in main
    args.oog, new_arg_modelhost, args.level)
File "/usr/local/lib/python3.6/dist-packages/metage2metabo/__main_analysis__.py", line␣
↪303, in main_analysis_workflow
    run_analysis_workflow(*allargs)
File "/usr/local/lib/python3.6/dist-packages/metage2metabo/m2m_analysis/m2m_analysis_
↪workflow.py", line 48, in run_analysis_workflow
    powergraph_analysis(gml_output, output_dir, oog_jar, taxon_file, taxonomy_level)
File "/usr/local/lib/python3.6/dist-packages/metage2metabo/m2m_analysis/graph_
↪compression.py", line 174, in powergraph_analysis
    merge_html_css_js(html_target +'_taxon', output_html_merged)
File "/usr/local/lib/python3.6/dist-packages/metage2metabo/m2m_analysis/graph_
↪compression.py", line 473, in merge_html_css_js
    cytoscape_min_js_str =  input_cytoscape_min_js.read()
File "/usr/lib/python3.6/encodings/ascii.py", line 26, in decode
    return codecs.ascii_decode(input, self.errors)[0]
UnicodeDecodeError: 'ascii' codec can't decode byte 0xe2 in position 234868: ordinal not␣
↪in range(128)
```

This is an issue with the encodign standard used. A solution is to set the local codec to *LC_ALL=C.UTF-8*:

```
LC_ALL=C.UTF-8 && srun singularity exec -B....
```

CHAPTER

THREE

ADDITIONAL FEATURES

M2M relies on packages that can also be used independantly with more features and additional options:

- mpwt: command-line and multi-process solutions to run Pathway Tools. Suitable to multiple reconstruction, for example genomes of a microbiota
- menetools: individual metabolic capabilities analysis using graph-based producibility criteria
- miscoto: community selection and metabolic screening in large-scal microbiotas, with or without taking a host into account

CHAPTER

FOUR

AUTHORS

Clémence Frioux and Arnaud Belcour, *Univ Bordeaux, Inria, INRAE, Bordeaux, France*, *Univ Grenoble Alpes, Inria, Grenoble, France* and *Univ Rennes, Inria, CNRS, IRISA, Rennes, France*.

CHAPTER

FIVE

ACKNOWLEDGEMENT

People of Pathway Tools (SRI International) for their help integrating Pathway Tools with command line and multi process in the mpwt package, used in M2M.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Ku2017] Kurtzer GM, Sochat V, Bauer MW (2017) Singularity: Scientific containers for mobility of compute. PLOS ONE 12(5): e0177459.

[Yo2003] Yoo, Andy B., Jette, Morris A., Grondona, Mark (2003). SLURM: Simple Linux Utility for Resource Management. Job Scheduling Strategies for Parallel Processing. Lecture Notes in Computer Science. 2862. p. 44.

[Eb2004] O. Ebenhöh, T. Handorf, and R. Heinrich, "Structural analysis of expanding metabolic networks.," Genome informatics. International Conference on Genome Informatics, vol. 15, no. 1, pp. 35–45, 2004.

PYTHON MODULE INDEX

m

INDEX

I

M

P

R

S

T

U