
MacSyFinder

Release 2.1.5

Sophie Abby, Bertrand Néron

Aug 20, 2025

CONTENTS

1	User Guide	3
1.1	User Guide	3
2	Modeller Guide	67
2.1	Modeller Guide	67
3	Developer Guide	99
3.1	Developer Guide	99
4	Indices and tables	113
	Python Module Index	115
	Index	117



Note

A **new version of MacSyFinder (v2)** is available, see [here for an overview of the novelties](#). The search engine was changed, and some bugs/unwanted behaviors corrected. MacSyFinder's models for v2 are very similar, yet not compatible with those from v1. See [here](#) for details on *how to carry your models to v2*.

The search engine of v2 being much different from that of v1, we **strongly suggest** to test whether the results are relevant by simply “translating” the models from v1 to v2, or if the models need to be adapted to correctly function with v2.

MacSyFinder is a program to **model and detect macromolecular systems, genetic pathways...** in protein datasets. In prokaryotes, these systems have often evolutionarily conserved properties:

- they are made of **conserved components**,
- they are encoded in **compact loci** (conserved genetic architecture).

The user models these systems with MacSyFinder to reflect these conserved features, and to allow their efficient detection.

Criteria for systems detection include **component content (quorum)**, and **genomic co-localization**. Each component corresponds to a hidden Markov model (HMM) protein profile to perform sequence similarity searches with the program Hmmer.

In order to model macromolecular systems, the user:

- builds or gather from databanks **HMM protein profiles** for components of interest,
- defines **decision rules** for each system in a dedicated XML grammar (see [Macromolecular models](#)).

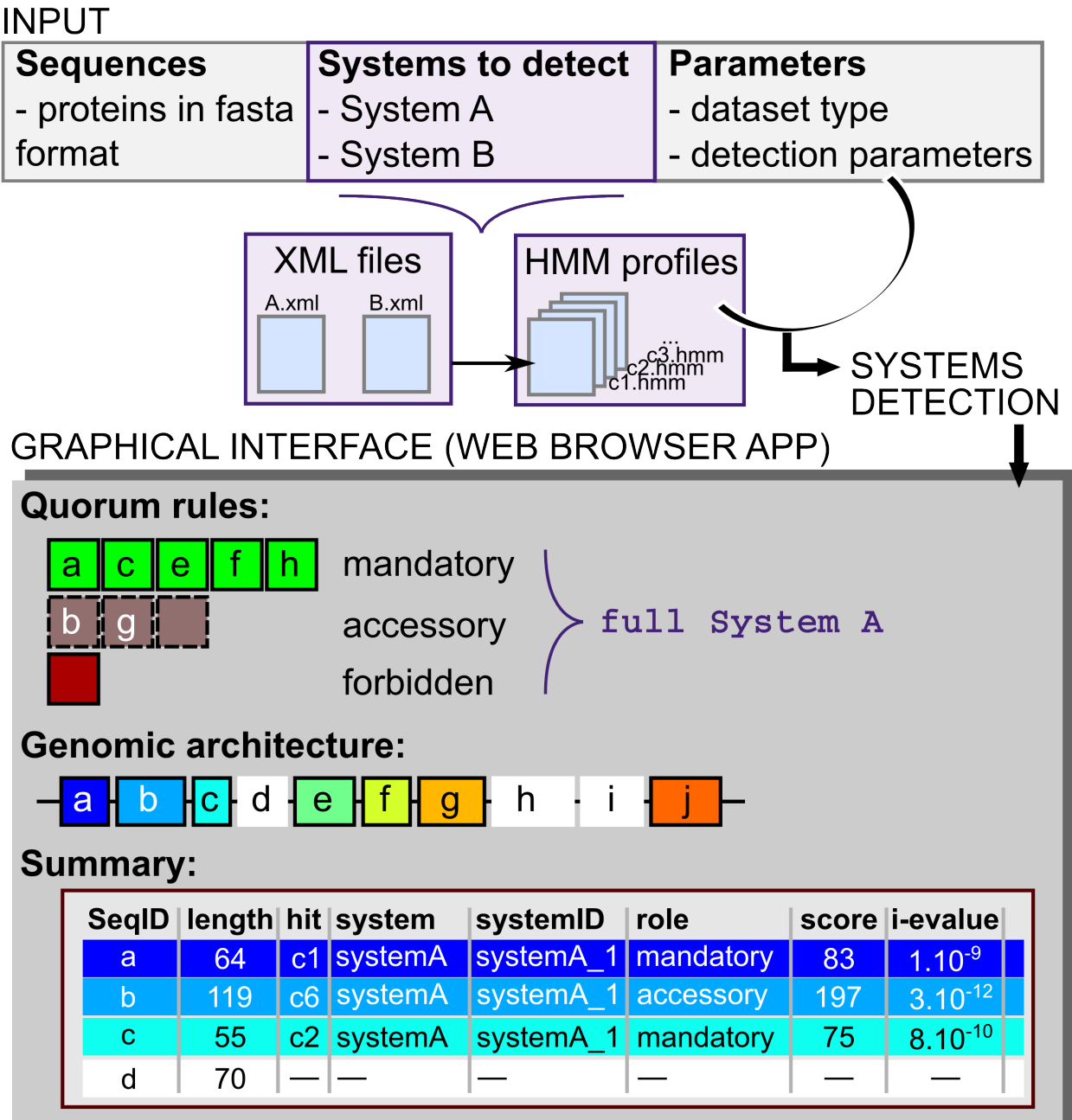
Note

If you use MacSyFinder v2, please cite:

Néron, Bertrand; Denise, Rémi; Coluzzi, Charles; Touchon, Marie; Rocha, Eduardo P.C.; Abby, Sophie S. MacSyFinder v2: Improved modelling and search engine to identify molecular systems in genomes. Peer Community Journal, Volume 3 (2023), article no. e28. doi : 10.24072/pcjournal.250. <https://peercommunityjournal.org/articles/10.24072/pcjournal.250/>

If you use MacSyFinder v1, please cite:

Abby SS, Néron B, Ménager H, Touchon M, Rocha EPC (2014). MacSyFinder: A Program to Mine Genomes for Molecular Systems with an Application to CRISPR-Cas Systems. PLoS ONE 9(10): e110726. doi:10.1371/journal.pone.0110726



USER GUIDE

1.1 User Guide

1.1.1 Running MacSyFinder

What's new in MacSyFinder v2?

V 2.1.5

MacSyFinder code has been refactored. Now there is a library MacSyLib that holds the most of the code. MacSyFinder holds only specific code, and relies on MacSyLib for the rest.

For users

The scripts have been renamed :

- *macsyprofile* become *msf_profile* (*macsyprofile* is deprecated and will be removed in future version)
- *macsydata* become *msf_data* (*macsyprofile* is deprecated and will be removed in future version)
- *macsy_merge* become *msf_merge* (*macsy_merge* is deprecated and will be removed in future version)
- *macsy_split* become *msf_split* (*macsy_split* is deprecated and will be removed in future version)
- *macsyfinder* stay unchanged.

msf_data has new subcommand *show*

It shows the structure of models available in an installed package of models. for instance:

```
$msf_data show TXSScan
TXSScan
├--archaea
│   └--Archaeal-T4P
├--bacteria
│   └--diderm
│       ├──Flagellum
│       ├──MSH
│       ├──T1SS
│       ├──T2SS
│       ├──T3SS
│       ├──T4aP
│       ├──T4bP
│       ├──T5aSS
│       └--T5bSS
```

(continues on next page)

(continued from previous page)

```
--T5cSS
--T6SSi
--T6SSii
--T6SSiii
--T9SS
--Tad
--pT4SSi
--pT4SSst
--monoderm
  --ComM
```

TXSScan (1.1.3) : 19 models

For modelers

- *msf_data check* now check profiles
 - if hmm file is not empty
 - if there is only one profile per hmm file

Use *msf_data check* before to publish your models. Also consider to add a pre-push git hook that run *msf_data check* when you push a tag. This hook is automatically installed when you run *msf_data init* to create a package skeleton (from V 2.1.4)

For developer

Rename the default branch from *master* to *main*

Most of the code has been moved in the MacSyLib package (<https://pypi.org/project/MacSyLib/>, <https://github.com/gem-pasteur/macsylib>). In MacSyFinder there is only the specific code and the entrypoints left.

V 2.1.4

For users

Minor bugs

- when profile name ends with *hmm*, then the profile was not retrieved by msf (<https://github.com/gem-pasteur/macsyfinder/issues/69>).
- fix omitted parameter *timeout* in *macsyfinder* step in *parallel_macsyfinder*

For modelers

- The *vers* is now deprecated in model/metadata file. *macsydata* rely only on the tag version to get the version of the model.
- *macsydata* now create a *git* repository for you and add files to it.
- a pre-push git hook is installed by *macsydata init* to prevent to push a tag (publish) a buggy model.

For developers

- all the *msf* code (macsypy and tests) pass the [ruff](#) linter
- all the code use type hints (required python >3.10) see (<https://github.com/gem-pasteur/macsyfinder/blob/main/CONTRIBUTING.md>)

V 2.1.3

Features

Support of *gzipped* files

MSF can also read *.gz* compressed files both for hmm profiles and sequences in fasta. It will uncompress them on the fly. The compressed files must end with the *.gz* extension. For the *hmmsearch* step You need to have *gunzip* installed on your system for this to work.

search engine

A group of hits that respect the distance constraints but each hit represent the same gene on the model, is **not** considered as a *cluster*.

A group of hits that respect the distance constraints but all hits represent a Neutral gene in model, is **not** considered as a *cluster*.

timeout improvement

If a replicon is skipped due to timeout during *best_solution* phase. The results corresponding to this replicon are not produced, but a warning indicating that *msf* skip this replicon appear in outputs.

V 2.1.1

Update MSF citation, fix minor bugs and add add few features

New features

–force option

Force MSF run even the out dir already exists and is not empty. Use this option with caution, MSF will erase everything in out dir before to run. <https://github.com/gem-pasteur/macsyfinder/issues/61>

Minor bugs

Macsyfinder with python subprocess kill main process on error

If an error occurred during HMM phase, all processes were killed as well the mother process but MSF stopped with an ugly traceback. <https://github.com/gem-pasteur/macsyfinder/issues/60>

In Gembase format parsing

The genes were not well grouped by contigs for draft genomes.

Cannot join current thread error during unit tests phase

Sometimes the testsuite failed with the following error: “cannot join current thread” <https://github.com/gem-pasteur/macsyfinder/issues/58>

V 2.1

Bug fix

Security patch

Patch macsydata to fix CVE-2007-4559 <https://github.com/gem-pasteur/macsyfinder/pull/57>

New features

Squash cluster of loners

If a cluster is made up with only loners, then the hits are treated by MSF as loners and not as regular cluster.

New option –timeout

In some case msf can take a long time to find the best solution (in ‘gembase’ and ‘ordered_replicon mode’). The timeout is per replicon. If this step reach the timeout, the replicon is skipped (for gembase mode the analyse of other replicons continue). NUMBER[SUFFIX] NUMBER seconds. SUFFIX may be ‘s’ for seconds (the default), ‘m’ for minutes, ‘h’ for hours or ‘d’ for days for instance 1h2m3s means 1 hour 2 min 3 sec. NUMBER must be an integer.

2.0

For Version 2, MacSyFinder was carried under [Python 3](#).

New features and search engine

MacSyFinder v2 is a major release. The **search engine** was changed for a more intuitive and comprehensive exploration of putative systems.

The search is now more thorough and avoid undesirable side-effects of the previous search engine. Being more thorough, it now also includes a **scoring scheme** to build candidate systems from sets of detected components (clusters), and can offer several optimal “solutions” (sets of detected systems) based on a combinatorial exploration of detected clusters. See [here for more details](#).

Warning

The search engine being different, one might want to check that models carried from v1 to v2 have the expected behaviour.

Several **new features** were added, including:

- a **new type of gene component** “neutral” was added in order to provide more possibilities for systems’ modelling in macsy-models. [See here](#) for more details.
- a **new component feature** was introduced: “multi-model”, that corresponds to components that are allowed to participate in occurrences of systems from different models. [See here](#) for more.
- more flexibility was introduced in the **search for systems’ components using HMMER**. It is now possible to use the *cut_ga* threshold when provided in the HMM profiles used for components’ similarity search. This enables to have a search tailored for each HMM profile, and thus component. [See here](#) for more details.

- a **new file structure** was created to better organize MacSyFinder’s packages (i.e. that include systems’ models and corresponding HMMER profiles). [See here](#) for details.
- a **tool** to easily install and distribute MacSyFinder’s packages was created. [See here](#) for more details on *macsy-data*.
- the **format for MacSyFinder’s models** has slightly changed, in order to offer more possibilities, and more readability. To see **how to carry models from v1 to v2**, [visit here](#).

Also, the search modes corresponding to “unordered” and “unordered_replicon” were merged into the “**unordered**” search mode - as they basically correspond to the same behaviour.

Note

In v2, output files were also re-defined. See [here for more details](#).

Dependencies

MacSyFinder v2 no longer requires the *formatdb* or *makeblastdb* tools from NCBI. However, new dependencies are used, but as they are Python libraries, it should be transparent for the user, and not require manual installations. See [here for details](#).

Models are more formalized

The models data are more formalized, with a well defined structure. For instance the definitions and profiles must be packed together in what we call a *macsy-model* package. If you intend to model new systems please refer to the [Modeller Guide](#).

Models installation

We now provide a new tool to manage the models. See [Models installation with msf_data](#).

Models configuration

The modeler can provide some specific configuration values released along the model package. See [Model configuration](#).

Modeller helper tool

To help modellers create new models we provide a new helper tool *macsyprofile*, which analyses HMMER raw output files from results of a previous MacSyFinder run, to provide information on all hits even if filtered out. See [msf_profile](#).

[Models installation with msf_data](#) provide also some options to help the modeller as

- **macsydata init** to init a new model package.
- **macsydata check** to check the integrity of a model package, before to use/publish it.

Installation

MacSyFinder works with models for macromolecular systems that are not shipped with it, you have to install them separately. See the [msf_data section](#) below. We also provide container so you can use macsyfinder directly.

MacSyFinder dependencies

Python version ≥ 3.10 is required to run MacSyFinder: <https://docs.python.org/3.10/index.html>

MacSyFinder has one program dependency:

- the *Hmmer* program, version 3.1 or greater (<http://hmmer.org/>).

The *hmmsearch* program should be installed (*e.g.*, in the PATH) in order to use MacSyFinder. Otherwise, the paths to this executable must be specified in the command-line: see the *command-line options*.

MacSyFinder also relies on six Python library dependencies:

- colorlog
- colorama
- pyyaml
- packaging
- networkx
- pandas

These dependencies will be automatically retrieved and installed when using *pip* for installation (see below).

Note

If you intend to build and distribute new models you will need some other dependencies see modeler guide for installation.

Note

If you want to contribute to the *MacSyFinder* code, check the guide lines (**CONTRIBUTING**) and specific procedure for *developer installation*.

MacSyFinder Installation procedure

It is recommended to use *pip* to install the MacSyFinder package.

Archive overview

- **doc** => The documentation in html and pdf
- **test** => All what is needed for unitary tests
- **macsypy** => The macsyfinder python library
- **setup.py** => The installation script
- **setup.cfg** => The installation script
- **pyproject.toml** => The project installation build tool
- **COPYING** => The licensing
- **COPYRIGHT** => The copyright
- **README.md** => Very brief macsyfinder overview
- **CONTRIBUTORS** => List of people who contributed to the code

- **CONTRIBUTING** => The guide lines to contribute to the code

Installation steps:

Make sure every required dependency/software is present.

By default MacSyFinder will try to use *hmmsearch* in your PATH. If *hmmsearch* is not in the PATH, you have to set the absolute path to *hmmsearch* in a *configuration file* or in the *command-line* upon execution. If the tools are not in the path, some test will be skipped and a warning will be raised.

Perform the installation.

```
python3 -m pip install macsyfinder
```

If you do not have the privileges to perform a system-wide installation, you can either install it in your home directory or use a *virtual environment*.

installation in your home directory

```
python3 -m pip install --user macsyfinder
```

installation in a virtualenv

```
python3 -m venv macsyfinder
cd macsyfinder
source bin/activate
python3 -m pip install macsyfinder
```

To exit the virtualenv just execute the *deactivate* command. To run *macsyfinder*, you need to activate the virtualenv:

```
source macsyfinder/bin/activate
```

Then run *macsyfinder* or *msf_data*.

Note

Super-user privileges (*i.e.*, *sudo*) are necessary if you want to install the program in the general file architecture.

Note

If you do not have the privileges, or if you do not want to install MacSyFinder in the Python libraries of your system, you can install MacSyFinder in a virtual environment (<http://www.virtualenv.org/>).

Warning

When installing a new version of MacSyFinder, do not forget to uninstall the previous version installed !

Uninstalling MacSyFinder

To uninstall MacSyFinder (the last version installed), run

```
(sudo) pip uninstall macsyfinder
```

If you install it in a virtualenv, just delete the virtual environment. For instance if you create a virtualenv name macsyfinder

```
python3 -m venv macsyfinder
```

To delete it, remove the directory

```
rm -R macsyfinder
```

From Conda/Mamba

From version 2.0, MacSyFinder is packaged for Conda/Mamba

```
mamba install -c macsyfinder=x.x
```

Where *x.x* is the macsyfinder version you want to install

From container

With Docker

The docker image is available on Docker Hub (<https://hub.docker.com/repository/docker/gempasteur/macsyfinder>) The computations are performed under *msf* user in */home/msf* inside the container. So You have to mount a directory from the host in the container to exchange data (inputs data, and results) from the host and the container. The shared directory must be writable by the *msf* user or overwrite the user in the container by your id (see example below)

Furthermore the models are no longer packaged along macsyfinder. So you have to install them by yourself. For that we provide a command line tool *msf_data* which is inspired by *pip*.

```
msf_data search PACKNAME
msf_data install PACKNAME== or >=, or ... VERSION
```

To work with Docker you have to install models in a directory which will be mounted in the image at run time

```
mkdir shared_dir
cd shared_dir
```

install desired models in *my_models* directory

```
docker run -v ${PWD}:/home/msf -u $(id -u ${USER}):$(id -g ${USER}) gempasteur/
↳macsyfinder:<tag> msf_data install --target /home/msf/my_models <MODELS_PACK>
```

run *msf* against all models contains in *<MODELS_PACK>*

```
docker run -v ${PWD}:/home/msf -u $(id -u ${USER}):$(id -g ${USER}) gempasteur/
↳macsyfinder:<tag> macsyfinder --db-type unordered_replicon --models-dir=/home/msf/my_
↳models/ --models <MODELS_PACK> all --sequence-db my_genome.fasta -w 12
```

With Apptainer (formerly Singularity)

As the docker image is registered in docker hub you can also use it directly with Apptainer (<https://apptainer.org/>). Unlike docker you have not to worry about shared directory, your HOME and /tmp are automatically shared.

```
# install desired models in my_models directory
apptainer run -H ${HOME} docker://gempasteur/macsyfinder:<tag> msf_data install --target_
↳ my_models <MODELS_PACK>

# run msf against all models contains in <MODELS_PACK>
apptainer run -H ${HOME} docker://gempasteur/macsyfinder:<tag> macsyfinder --db-type_
↳ unordered_replicon --models-dir=my_models --models <MODELS_PACK> all --sequence-db my_
↳ genome.fasta -w 12
```

If you intend to run *apptainer* from host which cannot access internet (cluster node for instance), you have to

1. download the image locally
2. transfert the image file on the right file system
3. and then use it.

```
apptainer build msf-<tag>.simg docker://gempasteur/macsyfinder:<tag>
cp msf-<tag>.simg <cluster_file_system>
apptainer run -H ${HOME} msf-<tag>.simg macsyfinder --db-type unordered_replicon --
↳ models-dir=my_models --models <MODELS_PACK> all --sequence-db my_genome.fasta -w 12
```

Models installation with *msf_data*

Once MacSyFinder is installed you have access to an utility program to manage the models: *msf_data*

This script allows to search, download, install and get information from MacSyFinder models stored on github (<https://github.com/macsy-models>) or locally installed. The general syntax for *msf_data* is:

```
msf_data <general options> <subcommand> <sub command options> <arguments>
```

To list all models available on *macsy-models*:

```
msf_data available
```

To search for models on *macsy-models*:

```
msf_data search TXSS
```

you can also search in models description:

```
msf_data search -S secretion
```

To install a model package:

```
msf_data install <model name>
```

To install a model when you have not the right to install it system-wide

To install it in your home (*./macsyfinder/data*):

```
msf_data install --user <model name>
```

To install it in any directory:

```
msf_data install --target <model dir> <model_name>
```

To know how to cite a model package:

```
msf_data cite <model name>
```

To show the name of the models and the structure of installed model package:

```
msf_data show <model package name>
```

for instance `msf_data show TXSScan`

```
TXSScan
├--archaea
│   └--Archaeal-T4P
└--bacteria
    ├──diderm
    │   ├──Flagellum
    │   ├──MSH
    │   ├──T1SS
    │   ├──T2SS
    │   ├──T3SS
    │   ├──T4aP
    │   ├──T4bP
    │   ├──T5aSS
    │   ├──T5bSS
    │   ├──T5cSS
    │   ├──T6SSi
    │   ├──T6SSii
    │   ├──T6SSiii
    │   ├──T9SS
    │   ├──Tad
    │   ├──pT4SSi
    │   └--pT4SSt
    └--monoderm
        └--ComM
```

TXSScan (1.1.3) : 19 models

To show the model definition:

```
msf_data definition <package or subpackage> model1 [model2, ...]
```

for instance to show model definitions T6SSii and T6SSiii in TXSS+/bacterial subpackage:

```
msf_data definition TXSS+/bacterial T6SSii T6SSiii
```

To show all models definitions in TXSS+/bacterial subpackage:

```
msf_data definition TXSS+/bacterial
```

To create a skeleton for your own model package (to access init subcommand check modeler installation):


```
msf_data init --pack-name <MY_PACK_NAME> --maintainer <"mantainer name"> --email
↪<maintainer email> --authors <"author1, author2, ..">
```

above `msf_data` with required options. Below I add optional but recommended options.

```
msf_data init --pack-name <MY_PACK_NAME> --maintainer <mantainer name> --email
↪<maintainer email> --authors <"author1, author2, .."> \
--license cc-by-nc-sa --holders <"the copyright holders"> --desc <"one line package_
↪description">
```

To list all `msf_data` subcommands:

```
msf_data --help
```

To list all available options for a subcommand:

```
msf_data <subcommand> --help
```

For models not stored in *macsy-models* the commands *available*, *search*, *installation* from remote or *upgrade* from remote are **NOT** available.

For models **NOT** stored in *macsy-models*, you have to manage them semi-manually. Download the archive (do not unarchive it), then use `msf_data` to install the archive.

MacSyFinder Quick Start

1. We recommend to install MacSyFinder using *pip* in a virtual environment (for further details see [Installation](#)).

```
python3 -m venv MacSyFinder
cd MacSyFinder
source bin/activate
pip install macsyfinder
```

Warning

hmmsearch from the HMMER package (<http://hmmer.org/>) must be installed.

2. Prepare your data. You need a file containing all protein sequences of your genome of interest in a FASTA file (for further details see [Input dataset](#)). In the best case scenario, they would be ordered as the corresponding genes are ordered along the replicons.
3. You need to install, or make available to MacSyFinder the models to search in your input genome data. Please refer to [Macromolecular models](#) to create your own package of models. Otherwise, macsy-models contributed by the community are available here: <https://github.com/macsy-models> and can be retrieved and installed using the `msf_data` command, installed as part of the MacSyFinder suite.

4. Command lines:

- Type: `macsyfinder -h`

To see all the options available. All command-line options are described in the [Command-line options section](#). In order to run MacSyFinder on your favorite dataset as soon as you have installed the macsy-model of interest, you can simply follow the following steps:

- Install the macsy-models of interest from the [Macsy Models repository](#):

```
msf_data install some-public-models
```

- On a “unordered” genome dataset:

```
macsyfinder --db-type unordered --sequence-db unordered_genome.fasta --models
model_family all
```

will search for systems corresponding to all the models of *model_family* modeled in .xml files shipped with the “*some-public-models*” macsy-model package, without taking into account the gene order.

- On a completely assembled genome (where the gene order is known):

```
macsyfinder --db-type ordered_replicon --sequence-db mygenome.fasta --models-dir
my-models --models model_family ModelA ModelB
```

will detect the macromolecular systems described in the two models “*ModelA*” and “*ModelB*” in a complete genome from the “*ModelA.xml*” and “*ModelB.xml*” definition files placed in the folder “*my-models/model_family/definitions*”.

- If you want to run the same analysis as above but with local macsy-models not installed by *msf_data*:

```
macsyfinder --db-type ordered_replicon --sequence-db mygenome.fasta --models-dir
my-models --models model_family ModelA ModelB
```

my-models is the directory containing the macsy-model packages. NB: The models must follow the *macsy-models package* structure.

Note

Systems names have to be spelled in a case-sensitive way to run their detection from the command-line. The name of the System corresponds to the suffix defined for xml files (.xml by default), for example “*toto*” for a model defined in “*toto.xml*”.

The “*all*” keyword allows to detect all models available in the definitions folder in a single run. See the *Command-line options*.

An example data set

We provide [here](#) an example dataset comprising a replicon and the output files expected with MacSyFinder, release 2.0 when running the TXSScan macsy-models. The genomic dataset consists in the complete sequence of chromosome I from *Vibrio cholerae* O1 biovar El Tor str. N16961 (published here: <https://pubmed.ncbi.nlm.nih.gov/10952301/>).

The chromosome to annotate is presented as a multi-FASTA file of the proteins ordered as the genes encoding them. An annotation of the protein secretion systems and appendages was run on the genome, using the macsyfinder set of models (“macsy-model”) TXSScan, V1.1.1 in the case of these examples. There are two output files offered, the one expected with the “ordered” genome mode of annotation, and the other with the “unordered” mode of genome annotation. The following command lines were used to obtain the output files:

1. The genome is downloaded from [here](#). It will serve as an input file in the next command-line examples.
2. The TXSScan models for annotation of secretion systems are installed. The command line is the following:

```
msf_data install TXSScan # Installs the latest version of TXSScan
```

3. MacSyFinder is run on the genome, here using 8 workers for the HMM search (“-w 8” option):
 - In “ordered” mode:

```
macsyfinder --sequence-db VICH001.B.00001.C001.fasta -o macsyfinder_TXSScan_VICH001_ordered
--models TXSScan all --db-type ordered_replicon -w 8 # specified output folder: mac-
syfinder_TXSScan_VICH001_ordered
```

- In “unordered” mode:

```
macsyfinder --sequence-db VICH001.B.00001.C001.fasta -o macsyfinder_TXSScan_VICH001_unordered
--models TXSScan all --db-type unordered -w 8 # specified output folder: mac-
syfinder_TXSScan_VICH001_unordered
```

The documentation on the generated output files can be consulted [here](#). See also our FAQ: *What search mode to be used?*

Note

A more comprehensive example of genome datasets with dedicated command lines and expected output files can be found [here](#).

Input and Options of MacSyFinder

Input dataset

The input dataset must be a set of protein sequences in **Fasta format** (see http://en.wikipedia.org/wiki/FASTA_format). (The fasta file can be compressed in *gzip* format see note below)

The *base section* in the configuration file (see *Configuration file*) can be used to specify **the path** and the **type of dataset** to deal with, as well as the *–sequence_db* and *–db_type* parameters respectively, described in the *Command-line options* (see *Input options*).

Four types of protein datasets are supported:

- *unordered* : a set of sequences corresponding to a complete genome (*e.g.* an unassembled complete genome)
- *ordered_replicon* : a set of sequences corresponding to an ordered complete replicon (*e.g.* an assembled complete genome)
- *gembase* : a set of multiple ordered replicons, which format follows the convention described in *Gembase format*.

For “ordered” (“ordered_replicon” or “gembase”) datasets only, MacSyFinder can take into account the **shape of the genome**: “linear”, or “circular” for detection. The default is set to “circular”.

This can be set with the *–replicon_topology* parameter from *Command-line options* (see *Input options*), or in the configuration in the *base section*.

With the “gembase” format, it is possible to specify a topology per replicon with a topology file (see *Gembase format* and *Topology files*).

Note

MSF can also read *.gz* compressed files; it will uncompress them on the fly. The compressed files must end with the *.gz* extension. For the *hmmsearch* step You need to have *gunzip* installed on your system for this to work.

Command-line options

Optional arguments:

```
-h, --help          Show the help message and exit

-m [MODELS [MODELS ...]], --models [MODELS [MODELS ...]]
                    The models to search. The --models option can be set several times.
↳ '
                    For each --models options the first element must be the name of
↳ family models,
                    followed by the name of the models.
                    If the name 'all' is in the list all models from the family will
↳ be searched.'
                    '--models TXSS Flagellum T2SS'
                        means MSF will search for models TXSS/Flagellum and TXSS/
↳ T2SS
                    '--models TXSS all'
                        means for all models found in model package TXSS
                    '--models CRISPRcas/subtyping all'
                        means MSF will search for all models described in the
↳ CRISPRCas/subtyping subfamily.
                    (required unless --previous-run is set)
```

Input dataset options:

```
--sequence-db SEQUENCE_DB
                    Path to the sequence dataset in fasta format.
                    (required unless --previous-run is set)
--db-type {ordered_replicon,gembase,unordered}
                    The type of dataset to deal with. "unordered" corresponds
                    to a non-assembled genome,
                    "ordered_replicon" to an assembled genome,
                    and "gembase" to a set of replicons where sequence identifiers
                    follow this convention: ">RepliconName_SequenceID".
                    (required unless --previous-run is set)
--replicon-topology {linear,circular}
                    The topology of the replicons
                    (this option is meaningful only if the db_type is
                    'ordered_replicon' or 'gembase'.
--topology-file TOPOLOGY_FILE
                    Topology file path. The topology file allows to specify a topology
                    (linear or circular) for each replicon (this option is meaningful
↳ only if
                    the db_type is 'ordered_replicon' or 'gembase'.
                    A topology file is a tabular file with two columns:
                    the 1st is the replicon name, and the 2nd the corresponding
↳ topology:
                    "RepliconA      linear"
--idx
                    Forces to build the indexes for the sequence dataset even
                    if they were previously computed and present at the dataset
↳ location.
                    (default: False)
```

Systems detection options:

```
--inter-gene-max-space INTER_GENE_MAX_SPACE INTER_GENE_MAX_SPACE
    Co-localization criterion: maximum number of components non-
    ↪matched by a
        profile allowed between two matched components
        for them to be considered contiguous.
        Option only meaningful for 'ordered' datasets.
        The first value must match to a model, the second to a number of ↪
    ↪components.
        This option can be repeated several times:
        "--inter-gene-max-space TXSS/T2SS 12 --inter-gene-max-space ↪
    ↪TXSS/Flagellum 20
--min-mandatory-genes-required MIN_MANDATORY_GENES_REQUIRED MIN_MANDATORY_GENES_REQUIRED
    The minimal number of mandatory genes required for model ↪
    ↪assessment.
        The first value must correspond to a model fully qualified name, ↪
    ↪the second value to an integer.
        This option can be repeated several times:
        "--min-mandatory-genes-required TXSS/T2SS 15 --min-mandatory-
    ↪genes-required TXSS/Flagellum 10"
--min-genes-required MIN_GENES_REQUIRED MIN_GENES_REQUIRED
    The minimal number of genes required for model assessment
    (includes both 'mandatory' and 'accessory' components).
    The first value must correspond to a model fully qualified name, ↪
    ↪the second value to an integer.
        This option can be repeated several times:
        "--min-genes-required TXSS/T2SS 15 --min-genes-required TXSS/
    ↪Flagellum 10
--max-nb-genes MAX_NB_GENES MAX_NB_GENES
    The maximal number of genes to consider a system as full.
    The first value must correspond to a model name, the second value ↪
    ↪to an integer.
        This option can be repeated several times:
        "--max-nb-genes TXSS/T2SS 5 --max-nb-genes TXSS/Flagellum 10"
--multi-loci MULTI_LOCI
    Specifies if the system can be detected as a 'scattered' system.
    The models are specified as a comma separated list of fully ↪
    ↪qualified name
        "--multi-loci model_familyA/model_1,model_familyB/model_2"
```

Options for Hmmer execution and hits filtering:

```
--hmmer HMMER          Path to the hmmsearch program.
                        If it is not specify rely on the PATH
                        (default: hmmsearch)
--e-value-search E_VALUE_SEARCH
    Maximal e-value for hits to be reported during hmmsearch search.
    By default MF set per profile threshold for hmmsearch run (--cut_
    ↪ga option)
        for profiles containing the GA bit score threshold.
        If a profile does not contains the GA bit score the --e-value-
    ↪search (-E in hmmsearch)
        is applied to this profile.
        To applied the --e-value-search to all profiles use the --no-cut-
```

(continues on next page)

(continued from previous page)

```

↪ga option.
                                (default: 0.1)
--no-cut-ga                      By default the MSF try to applied a threshold per profile by using
↪the
                                hmmer -cut-ga option. This is possible only if the GA bit score is
↪present in the profile otherwise
                                MF switch to use the --e-value-search (-E in hmmsearch).
                                If this option is set the --e-value-search option is used for all
↪profiles regardless the presence of
                                the a GA bit score in the profiles.
                                (default: False)
--cut-ga                         By default the MSF try to applied a threshold per profile by using
↪the
                                hmmer -cut-ga option. This is possible only if the GA bit score is
↪present in the profile otherwise
                                MSF switch to use the --e-value-search (-E in hmmsearch).
                                But the modeler can override this default behavior to do not use
↪cut_ga but --e-value-search instead (-E in hmmsearch).
                                The user can reestablish the general MSF behavior, be sure the
↪profiles contain the GA bit score.
                                (default: True)

--i-evalue-sel I_EVALUE_SEL
                                Maximal independent e-value for Hmmer hits to be selected for
↪system detection.
                                (default:0.001)
--coverage-profile COVERAGE_PROFILE
                                Minimal profile coverage required in the hit alignment to allow
                                the hit selection for system detection.
                                (default: 0.5)

```

Options for clusters and systems' scoring:

```

--mandatory-weight MANDATORY_WEIGHT
                                the weight (score) of a mandatory component when scoring clusters
                                (default:1.0)
--accessory-weight ACCESSORY_WEIGHT
                                the weight (score) of an accessory component when scoring clusters
                                (default:0.5)
--exchangeable-weight EXCHANGEABLE_WEIGHT
                                the weight modifier for the score of a component that is
↪exchangeable
                                (default:0.8)
--redundancy-penalty REDUNDANCY_PENALTY
                                the weight modifier for the score of a component that is already
↪present in another cluster
                                (default:1.5)

--loner-multi-system-weight LONER_MULTI_SYSTEM_WEIGHT
                                the weight modifier for the score of a component that is `loner`
↪and `multi-system` at the same time
                                (default:0.7)

```

Path options:

```
--models-dir MODELS_DIR
    specify the path to the models if the models are not installed in
    ↳ the canonical place.
    It gather definitions (xml files) and hmm profiles in a specific
    structure. A directory with the name of the model with at least
    ↳ two directories
    ↳ definitions and
    ↳ subdirectories
    profiles" which contains all hmm profile for gene describe in
    models" which contains either xml file of definitions or
    to organize the model in subsystems.
-o OUT_DIR, --out-dir OUT_DIR
    Path to the directory where to store results.
    if out-dir is specified res-search-dir will be ignored.
--force
    force to run even the out dir already exists and is not empty.
    Use this option with caution, MSF will erase everything in out dir
    ↳ before to run.
--index-dir INDEX_DIR
    Specifies the path to a directory to store/read the sequence index
    ↳ when the sequence-db dir
    is not writable.
--res-search-suffix RES_SEARCH_SUFFIX
    The suffix to give to Hmmer raw output files. (default: .search_
    ↳ hmm.out)
--res-extract-suffix RES_EXTRACT_SUFFIX
    The suffix to give to filtered hits output files. (default: .res_
    ↳ hmm_extract)
--profile-suffix PROFILE_SUFFIX
    The suffix of profile files. For each 'Gene' element, the
    ↳ corresponding profile is
    searched in the 'profile_dir', in a file which name is based on the
    Gene name + the profile suffix.
    For instance, if the Gene is named 'gspG' and the suffix is '.hmm3
    ↳ ',
    then the profile should be placed at the specified location
    and be named 'gspG.hmm3'
    (default: .hmm)
```

General options:

```
-w WORKER, --worker WORKER
    Number of workers to be used by MacSyFinder.
    In the case the user wants to run MacSyFinder in a multi-thread
    ↳ mode.
    (0 mean all threads available will be used).
    (default: 1)
-v, --verbosity
    Increases the verbosity level. There are 4 levels:
    Error messages (default), Warning (-v), Info (-vv) and Debug.(-
    ↳ vv)
--mute
    mute the log on stdout.
    (continue to log on macsyfinder.log)
    (default: False)
```

(continues on next page)

(continued from previous page)

```

--version          show program's version number and exit
-l, --list-models  display the all models installed in generic location and quit.
--cfg-file CFG_FILE Path to a MacSyFinder configuration file to be used.
--previous-run PREVIOUS_RUN
                    Path to a previous MacSyFinder run directory.
                    It allows to skip the Hmmer search step on same dataset,
                    as it uses previous run results and thus parameters regarding
↳ Hmmer detection.
                    The configuration file from this previous run will be used.
                    Conflict with options
                        --config, --sequence-db, --profile-suffix, --res-extract-
↳ suffix, --e-value-res, --db-type, --hmmer
--timeout TIMEOUT  In some case msf can take a long time to find the best solution,
↳ (in 'gembase' and 'ordered_replicon mode').
                    The timeout is per replicon. If this step reach the timeout, the
↳ replicon is skipped (for gembase mode the analyse of other replicons continue).
                    NUMBER[SUFFIX] NUMBER seconds. SUFFIX may be 's' for seconds (the
↳ default), 'm' for minutes, 'h' for hours or 'd' for days
                    for instance 1h2m3s means 1 hour 2 min 3 sec. NUMBER must be an
↳ integer.

```

Note

For some command line examples, have a look [here](#), or at the *MacSyFinder Quick Start* section.

Configuration file

Options to run MacSyFinder can be specified in a configuration file.

A macyfinder utility is provided to generate macyfinder config file: *msf_config*

msf_config is a conversation menu which guide you and generate a file *macyfinder.conf* in ini format. Once generated put this file in specific locations (see below) to be take in account by MacSyFinder.

The Config object handles all configuration options for MacSyFinder (for more details read [Config object in MacSyLib documentation](#)). There kind of locations where to put configuration file:

1. System wide configuration (this configuration is used for all macyfinder run)
 - */etc/macyfinder/macyfinder.conf*
 - or in *\${VIRTUAL_ENV}/etc/macyfinder.conf* if you installed macyfinder in a virtualenv
 - the file pointed by environment variable *MACSY_HOME*
2. User wide configuration (this configuration is used for all run for a user)
 - *~/.macyfinder/macyfinder.conf*
3. Project configuration
 - *macyfinder.conf* in the current directory
 - with command line option *-cfg-file*

Note

The precedence rules from the least to the most important priority are:

System wide configuration < user wide configuration < project configuration < command line option

This means that command-line options will always bypass those from the configuration files. In the same flavor, options altering the definition of systems found in the command-line or the configuration file will always overwhelm values from systems' *XML definition files*.

The configuration files must follow the Python “ini” file syntax. The Config object provides some default values and performs some validations of the values.

In MacSyFinder, six sections are defined and stored by default in the configuration file:

- **base** : all information related to the protein dataset under study
 - *sequence_db* : the path to the dataset in Fasta format (*no default value*)
 - *db_type* : the type of dataset to handle, four types are supported:
 - * *unordered* : a set of sequences corresponding to a complete replicon (*e.g.* an unassembled complete genome)
 - * *ordered_replicon* : a set of sequences corresponding to a complete replicon ordered (*e.g.* an assembled complete genome)
 - * *gembase* : a set of multiple ordered replicons.*(no default value)*
 - *replicon_topology* : the topology of the replicon under study. Two topologies are supported: ‘linear’ and ‘circular’ (*default* = ‘circular’). This option will be ignored if the dataset type is not ordered (*i.e.* “unordered_replicon” or “unordered”).
- **models** * list of models to search in replicon
- **models_opt**
 - *inter_gene_max_space* = list of models’ fully qualified names and integer separated by spaces (see example below). These values will supersede the values found in the model definition file.
 - *min_mandatory_genes_required* = list of models’ fully qualified name and integer separated by spaces. These values will supersede the values found in the model definition file.
 - *min_genes_required* = list of models’ fully qualified name and integer separated by spaces. These values will supersede the values found in the model definition file.
 - *max_nb_genes* = list of models’ fully qualified names and integer separated by spaces. These values will supersede the values found in the model definition file.
- **hmmer**
 - *hmmer_exe* (default= *hmmsearch*)
 - *e_value_res* = (default= *1*)
 - *i_value_sel* = (default= *0.5*)
 - *coverage_profile* = (default= *0.5*)
- **score_opt**
 - *mandatory_weight* (default= *1.0*)

- *accessory_weight* (default= 0.5)
- *exchangeable_weight* (default= 0.8)
- *redundancy_penalty* (default= 1.5)
- *out_of_cluster* (default= 0.7)

- **directories**

- *res_search_dir* = (default= ./datatest/res_search)
- *res_search_suffix* = (default= .search_hmm.out)
- *system_models_dir* = (default= ./models)
- *res_extract_suffix* = (default= .res_hmm_extract)
- *index_dir* = (default= beside the sequence_db)

- **general**

–*log_level*: (default= *debug*) This corresponds to an integer code:

Level	Numeric value
CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10
NOTSET	0

- *log_file* = (default = macsyfinder.log in directory of the run)

Example of a configuration file

```
[base]
prefix = /path/to/macsyfinder/home/
file = %(prefix)s/data/base/prru_psae.001.c01.fasta
db_type = gembase
replicon_topology = circular

[models]
models_1 = TFF-SF_final all

[models_opt]
inter_gene_max_space = TXSS/T2SS 22 TXSS/Flagellum 44
min_mandatory_genes_required = TXSS/T2SS 6 TXSS/Flagellum 4
min_genes_required = TXSS/T2SS 8 TXSS/Flagellum 4
max_nb_genes = TXSS/T2SS 12 TXSS/Flagellum 8

[hmmmer]
hmmmer = hmmsearch
e_value_res = 1
i_evalue_sel = 0.5
coverage_profile = 0.5

[score_opt]
```

(continues on next page)

(continued from previous page)

```

mandatory_weight = 1.0
accessory_weight = 0.5
exchangeable_weight = 0.8
redundancy_penalty = 1.5
loner_multi_system_weight = 0.7

[directories]
prefix = /path/to/macsyfinder/home/
data_dir = %(prefix)s/data/
res_search_dir = %(prefix)s/dataset/res_search/
res_search_suffix = .raw_hmm
system_models_dir = %(data_dir)/data/models, ~/.macsyfinder/data
profile_suffix = .fasta-aln.hmm
res_extract_suffix = .res_hmm
index_dir = path/where/I/store/my_indexes

[general]
log_level = debug
worker = 4

```

Note

After a run, the corresponding configuration file (“macsyfinder.conf”) is generated as a (re-usable) output file that stores every options used in the run. It is stored in the results’ directory (see [the output section](#)).

Warning

The configuration variable *models_dir* cannot be set in general configuration file. *models_dir* can be set only in configuration under user control. `$(HOME)/.macsyfinder/macsyfinder.conf < macsyfinder.conf < "command-line" options` *models_dir* is a single path to a directory where macsyfinder can find models.

But the *system_models_dir* can be set in general configuration file

- /etc/macsyfinder/macsyfinder.conf
- or \${VIRTUAL_ENV}/etc/macsyfinder/macsyfinder.conf
- or anywhere point by \$MACSY_CONF environment variable

system_models_dir manage a list of locations where macsyfinder can find models. The order of locations is important, it reflects the precedence rule (The models found in last location superseed models found in previous location). By default look for following directories: `/share/macsyfinder/models`, or `/usr/share/macsyfinder/models` and `$(HOME)/macsyfinder/models` and *system_models_dir* uses these directories if they exists.

In-house input files**Gembase format**

In order to allow the users to run MacSyFinder on **several genomes at once**, we propose to adopt the following convention to fulfill the requirements for the “gembase db_type”.

It consists in providing for each protein, both the replicon name and a protein identifier separated by a “_” in the first field of fasta headers. “_” are accepted in the replicon name, but not in the protein identifier. Hence, the last “_” is

the separator between the replicon name and the protein identifier. As such, MacSyFinder will be able to treat each replicon separately to assess macromolecular systems' presence.

For instance:

```
>PlasmidA_0001 YP_003225072.1 | putative stcE protein
MKLKYLSCMILASLAMGAFAATAADNNSAIYFNTTQPVNDLQGGLAEEVK
FAQSQILSAHPKEGESQQHLTSLRKSLLLVRLVKADDKTPVQVEARDAND
KILGTLTSPSSLPDTPVYHLDGVPADGIDFTPQNGTKKIINTVAEVNKL
SDASGSSIKSYLANALVEIQTANGRWIRDMYLPQGAEEGKMVRVFSYA
GYNSTVFYGRDKVTL SVGNTLLFKYVNGQWFRSGELENNRIAYAQHTWSA
ELPAHWIVPGLNLVIKQGNLSGSLNDINVGAPGELLHTIDIGMLTTPRG
RFDFAKDKEAHREYFQTIPVSRMIVNNYAPLHLKEVMLPTGTLLTDADPG
>PlasmidA_0002 YP_003225073.1 | type II secretion protein EtpC
MLFFLSSRRDRNLFIKDIALKMLTPNWVLCVILLIAGYQLVSVIRHFWLT
PATASDLSHVSSETAVTDEHTEENFVFTLFGTASPPLSEGKVQKTTSS
LSDDLSSGGDLVDRGILYSSVTEHSVAIFAHNNRQFSLGIGEKVPGYDAT
ISAIKSDHIVINYQGNASLPLRYDNPAPKRNAQDDNLLIVGPVTTQANFR
VKNIFDMSLSPVTVNNTLSGYRLSPGKASSLFYNAGLHDNDLAVLLNGS
ELRDRTRQAKQIMKQLTELKEIKITVERDQGLYDAFIAVGEN
....
>ChromosomeA_0001 YP_003573410.1 | adhesin-like protein
MKKLFLFAALLMTGFAYSCEDVVDNPAQDPAQSWNYSVSVKFADFDFNG
AVDENSVPYTYKAPTTLYVLNEENTLMGTITTTDAAPAIGDYGTAGTLTG
SIGNNLIITTKIGNDLTKQDGLKSAIENGIVQTAEVPIKIYNANSGLT
TASAKMDNTAAIAYTSLGYIKGGDKILFVEGNQTFEWTVNEEFDPTSTD
LYIALPMNTDPETEYTISSDSKDGTRGGTFKLADYPTLAAGKVSNIYIGG
IPFIQTGVDLTKWDAYMRDPNNTWYMNINNGWPATFSQEVEDGKSFIV
TQSGPTLDSLNVVGGVTGKEVNTLNNIRLGKDRSINIGDKHGWVEYDG
THDIYGWGAKANVTLIGENECETLYIQCPATKKGEGTLNKNLSIDSYGS
>ChromosomeA_0020 YP_003573411.1 | hypothetical protein
MKRIVLITLVSILTTFFQAIQVANGFYRVQNNASSRYITLRDNAVGTVDY
SSTNVDSLNIWTSWGFVKSNPASIIYVEQHDSKYDLKVQGTGIYAITG
GRTYLELRPKDSGYILAVTYNGMEGRLYDSEEDVDGEGYVKRSGNSAYQY
WSFIPVDTENNYIGLQPTVQVGDNYYGTLYASYPFKAASSGIKFYYVDAI
....
>NC_001548_0015 YP_003225080.1 | type II secretion protein EtpJ (translation)
MSQQRVKGFLLLEMLLALAVFAALSISAFQVLQSGIRAHLSQDKVRRLA
ELQRGGSQIERDLMQMIHRHSGSEGLLLAAPHLLKSDDWGISFTRNSWL
NPAGMLPRPELQWVGYYRLRQKLERLSYFYVDHPSGIAPDVRVVLGVHA
FRLRFFVNGTWQARWDSTSILPQAVEVTLVMDDFAELTRLFLVSKETAE
```

This input file contains 3 replicons: PlasmidA (which 2 first protein identifiers are 0001 and 0002), ChromosomeA (which 2 first protein identifiers are 0001 and 0020) and NC_001548 (which first protein identifier is 0015). MacSyFinder search results will thus be reported for each of these three replicons.

Warning

This *gembase* format is old and not compliant with the *gembase* format produced by [PanACoTA](#). The support of the new *gembase* format is in the road map.

Topology files

To be able to attribute a topology per replicon/genome when using the Gembase format, we propose the user to build a “topology file” in the form of a tabular file with two columns separated by a “:”. The 1st column is the replicon name, and the 2nd the corresponding topology. Comments can be written after a “#”.

For example:

```
# comment line
PlasmidA : circular
ChromosomeA : linear
ChromosomeB : circular
```

Note

A topology file can be specified on the command-line with the `--topology-file` parameter.

Output format

MacSyFinder provides different types of output files. At each run, MacSyFinder creates a new folder, whose name is based on a fixed prefix and a random suffix, for instance “macsyfinder-20130128_08-57-46”. MacSyFinder output files are stored in this run-specific folder.

There are three types of output files:

1. The main output files for the systems’ search. They differ with the search mode (*ordered* or *unordered*).
2. The *HMMER output files* (search of each systems’ components), located in the *hmm_results* folder.
3. The internal *configuration and log files*.

Note

Each tabular output file contains a header line describing each column in the output.

Output files for the “ordered replicon(s)” search modes

These output files are provided when MacSyFinder search proceeds on a set of proteins that are deemed to follow the order of their genes on replicons. This corresponds to the two search modes *gembase* and *ordered_replicon*.

Systems detection results

Different types of output files are provided, human-readable files “.txt”, and tabulated files “.tsv”. For the latter, headers are provided with the content of the lines in the file.

- *best_solution.tsv* - This file contains the **best solution found by MacSyFinder** in terms of systems detected, under the form of a per-component, tabulated report file. A **solution** consists in a set of compatible systems (no components’ overlap allowed). If multiple solutions showed a maximal score, a *ranking* is established.

To see potential other best solutions (in case several obtained the same highest score), see file *all_best_solutions.tsv*.

To see all possible, candidate systems without further processing, see files *all_systems.txt* and *all_systems.tsv*.

The *best_solution.tsv* file is the most similar to former V1 file *macsyfinder.report*.

- *best_solution_loners.tsv* and *best_solution_multisystems.tsv* report hits which have been identified as loners or multi-systems which means that the corresponding gene is tagged as a 'loner' or 'multi-system' in the model definition and the hit is located in a cluster.
- *best_solution_summary.tsv* is a summary of the *best_solution.tsv* file, containing the number of systems detected in each replicon analysed.
- *all_systems.txt* - This file describes the search process of all possible candidate systems given the definitions in systems' models - without processing of the potential overlaps between candidate systems. This set of possible candidate systems are also given under the form of a tabulated file in *all_systems.tsv*.
- *rejected_candidates.tsv* and *rejected_candidates.txt* - This file lists candidate clusters (or a combination of clusters) components that were rejected by MacSyFinder during the search process, and were thus not assigned to a candidate system. This set of clusters are also given under the form of tabulated file *rejected_candidates.tsv*.
- *all_best_solutions.tsv* - This file contains all possible best solutions under the form of a per-component, tabulated report file. To retrieve a single best solution as proposed by MacSyFinder, see file *best_solution.tsv*.
- *all_systems.tsv* - This file contains all possible candidate systems given the definitions - without processing of the potential overlaps between candidate systems, under the form of a per-component, tabulated report file. It corresponds to the tabulated version of the *all_systems.txt* file.

all_systems.txt

The file starts with some comments:

- the version of MacSyFinder used
- the name of model package and version used
- the command line used to produce this file

Then for each replicon, the systems detected are listed along with their description:

- **system_id** - the unique identifier of a system
- **model** - the model assigned to this system
- **replicon** - the name of the replicon harbouring the system
- **clusters** - the clusters composition of this system
 - each clusters is a list of tuple
 - each tuple is composed of:
 - * the name of the matching gene(s) in the replicon
 - * the name of the corresponding gene profile(s)
 - * the position of the corresponding sequence(s) along the replicon
- **occurrence** - the average number of occurrences of each components of the system (as a potential proxy to estimate whether there's the genetic potential for multiple systems in one)
- **wholeness** - the percentage of the model's components that were found in this system
- **loci nb** - the number of different loci constituting this system
- **score** - the score of the system. See [here](#) for more details
- **systems components** - the number of occurrences of each model components in parenthesis the name of the matching profile in square brackets the name of other putative systems that would involve this gene

Here is an example of the *all_systems.txt* file:

```

# macsyfinder 20200217.dev
# models: TFF-SF_final-0.1
# macsyfinder --sequence-db DATA_TEST/sequences.prt --db-type=gembase --models-dir data/
↳models/ --models TFF-SF_final all -w 4
# Systems found:

system id = VICH001.B.00001.C001_MSH_1
model = TFF-SF_final/MSH
replicon = VICH001.B.00001.C001
clusters = [('VICH001.B.00001.C001_00406', 'MSH_mshI', 366), ('VICH001.B.00001.C001_00407
↳', 'MSH_mshJ', 367), ('VICH001.B.00001.C001_00408', 'MSH_mshK', 368), ('VICH001.B.
↳00001.C001_00409', '
MSH_mshL', 369), ('VICH001.B.00001.C001_00410', 'MSH_mshM', 370), ('VICH001.B.00001.C001_
↳00411', 'MSH_mshN', 371), ('VICH001.B.00001.C001_00412', 'MSH_mshE', 372), ('VICH001.B.
↳00001.C001_0041
3', 'MSH_mshG', 373), ('VICH001.B.00001.C001_00414', 'MSH_mshF', 374), ('VICH001.B.00001.
↳C001_00415', 'MSH_mshB', 375), ('VICH001.B.00001.C001_00416', 'MSH_mshA', 376), (
↳'VICH001.B.00001.C001
_00417', 'MSH_mshC', 377), ('VICH001.B.00001.C001_00418', 'MSH_mshD', 378), ('VICH001.B.
↳00001.C001_00419', 'MSH_mshO', 379), ('VICH001.B.00001.C001_00420', 'MSH_mshP', 380), (
↳'VICH001.B.00001
.C001_00421', 'MSH_mshQ', 381)]
occ = 1
wholeness = 0.941
loci nb = 1
score = 10.500

mandatory genes:
  - MSH_mshA: 1 (MSH_mshA)
  - MSH_mshE: 1 (MSH_mshE)
  - MSH_mshG: 1 (MSH_mshG)
  - MSH_mshL: 1 (MSH_mshL)
  - MSH_mshM: 1 (MSH_mshM)

accessory genes:
  - MSH_mshB: 1 (MSH_mshB)
  - MSH_mshC: 1 (MSH_mshC)
  - MSH_mshD: 1 (MSH_mshD)
  - MSH_mshF: 1 (MSH_mshF)
  - MSH_mshI: 1 (MSH_mshI)
  - MSH_mshI2: 0 ()
  - MSH_mshJ: 1 (MSH_mshJ)
  - MSH_mshK: 1 (MSH_mshK)
  - MSH_mshN: 1 (MSH_mshN)
  - MSH_mshO: 1 (MSH_mshO)
  - MSH_mshQ: 1 (MSH_mshQ)
  - MSH_mshP: 1 (MSH_mshP)

neutral genes:

=====
system id = VICH001.B.00001.C001_T4P_14
model = TFF-SF_final/T4P

```

(continues on next page)

(continued from previous page)

```

replicon = VICH001.B.00001.C001
clusters = [('VICH001.B.00001.C001_00476', 'T4P_pilT', 427), ('VICH001.B.00001.C001_00477
→', 'T4P_pilU', 428)], [('VICH001.B.00001.C001_00847', 'T4P_pilO', 778), ('VICH001.B.
→00001.C001_00850',
→ 'T4P_pilE', 781), ('VICH001.B.00001.C001_00851', 'T4P_fimT', 782), ('VICH001.B.00001.
→C001_00852', 'T4P_pilW', 783), ('VICH001.B.00001.C001_00853', 'T4P_pilX', 784), (
→ 'VICH001.B.00001.C001_00
854', 'T4P_pilV', 785)], [('VICH001.B.00001.C001_02305', 'T4P_pilA', 2202), ('VICH001.B.
→00001.C001_02306', 'T4P_pilB', 2203), ('VICH001.B.00001.C001_02307', 'T4P_pilC', 2204),
→ ('VICH001.B.000
01.C001_02308', 'T4P_pilD', 2205)], [('VICH001.B.00001.C001_02502', 'MSH_mshM', 2391), (
→ 'VICH001.B.00001.C001_02505', 'T4P_pilQ', 2394), ('VICH001.B.00001.C001_02506', 'T4P_
→pilP', 2395), ('VI
CH001.B.00001.C001_02507', 'T4P_pilO', 2396), ('VICH001.B.00001.C001_02508', 'T4P_pilN',
→2397), ('VICH001.B.00001.C001_02509', 'T4P_pilM', 2398)]
occ = 1
wholeness = 0.944
loci nb = 4
score = 12.000

mandatory genes:
    - T4P_pilE: 1 (T4P_pilE)
    - T4P_pilB: 1 (T4P_pilB)
    - T4P_pilC: 1 (T4P_pilC)
    - T4P_pilO: 2 (T4P_pilO, T4P_pilO)
    - T4P_pilQ: 1 (T4P_pilQ)
    - T4P_pilN: 1 (T4P_pilN)
    - T4P_pilT: 1 (T4P_pilT)
    - T4P_pilD: 1 (T4P_pilD [VICH001.B.00001.C001_T2SS_4])

accessory genes:
    - T4P_pilA: 1 (T4P_pilA)
    - T4P_pilV: 1 (T4P_pilV)
    - T4P_pilY: 0 ()
    - T4P_pilW: 1 (T4P_pilW)
    - T4P_pilX: 1 (T4P_pilX)
    - T4P_fimT: 1 (T4P_fimT)
    - T4P_pilM: 1 (T4P_pilM)
    - T4P_pilP: 1 (T4P_pilP)
    - T4P_pilU: 1 (T4P_pilU)
    - MSH_mshM: 1 (MSH_mshM)

neutral genes:

```

all_systems.tsv

This corresponds to the tabulated version of the systems listed in *all_systems.txt*. Each line corresponds to a “hit” that has been assigned to a detected system. It includes:

- **replicon** - the name of the replicon it belongs to
- **hit_id** - the unique identifier of the hit
- **gene_name** - the name of the component identified by the hit

- **hit_pos** - the position of the sequence in the replicon
- **model_fqn** - the model fully-qualified name
- **sys_id** - the unique identifier attributed to the detected system
- **sys_loci** - the number of loci
- **locus_num** - the number of the locus where is located this gene. Loners gene have a negative locus_num
- **sys_wholeness** - the wholeness of the system
- **sys_score** - the system score
- **sys_occ** - the estimated number of system occurrences that could be potentially “filled” with this system’s occurrence, based on the average number of each component found. A proxy for the genetic potential to encode several systems from the set of components found in this one occurrence.
- **hit_gene_ref** - the gene in the model whose this hit plays the role of
- **hit_status** - the status of the component in the assigned system’s definition
- **hit_seq_len** - the length of the protein sequence matched by this hit
- **hit_i_eval** - Hmmer statistics, the independent-evalue
- **hit_score** - Hmmer score
- **hit_profile_cov** - the percentage of the profile covered by the alignment with the sequence
- **hit_seq_cov** - the percentage of the sequence covered by the alignment with the profile
- **hit_begin_match** - the position in the sequence where the profile match begins
- **hit_end_match** - the position in the sequence where the profile match ends
- **counterpart** - the hit id of some other hit which are equivalent. Only loners and multi-systems hits have counterparts
- **used_in** - whether the hit could be used in another system’s occurrence

This file can be easily parsed using the Python [pandas](#) library.

```
import pandas as pd

systems = pd.read_csv("path/to/systems.tsv", sep='\t', comment='#')
```

Note

Each system reported is separated from the others with a blank line to ease human reading. These lines are ignored during the parsing with pandas.

```
# macsyfinder 20220121.dev
# models : functional-0.0b2
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsyfinder --db-
→type=gembase --models-dir=tests/data//models/ --models TFF-SF Archaeal-T4P ComM MSH_
→T2SS T4bP T4P Tad --relative-path --sequence-db tests/data/base/gembase.fasta -w 12
# Systems found:
replicon      hit_id      gene_name      hit_pos      model_fqn      sys_
→id          sys_loci      locus_num      sys_wholeness      sys_score      sys_
→occ          hit_gene_ref      hit_status      hit_seq_len      hit_i_
```

(continues on next page)

(continued from previous page)

→eval	hit_score	hit_profile_cov	hit_seq_cov	hit_begin_
→match	hit_end_match	counterpart	used_in	
GCF_000005845	GCF_000005845_000970	T4P_pilC	97	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	1	0.556
→260	1	T4P_pilC	400	2.2e-105
→100	0.991	0.830	62	353.
GCF_000005845	GCF_000005845_000980	T4P_pilB	98	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	1	0.556
→260	1	T4P_pilB	461	8.9e-152
→100	0.948	0.850	62	506.
GCF_000005845	GCF_000005845_000990	T4P_pilA	99	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	1	0.556
→260	1	T4P_pilA	146	1.1e-19
→200	0.859	0.473	5	71.
GCF_000005845	GCF_000005845_025680	T4P_pilW	2568	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	2	0.556
→260	1	T4P_pilW	187	3.3e-08
→500	0.625	0.401	6	34.
GCF_000005845	GCF_000005845_025690	T4P_fimT	2569	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	2	0.556
→260	1	T4P_fimT	156	2.5e-06
→500	0.939	0.397	5	28.
GCF_000005845	GCF_000005845_030590	T4P_pilQ	3059	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	3	0.556
→260	1	T4P_pilQ	412	5.9e-51
→100	0.919	0.408	244	173.
GCF_000005845	GCF_000005845_030620	T4P_pilN	3062	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	3	0.556
→260	1	T4P_pilN	179	3.8e-09
→500	0.986	0.765	5	37.
GCF_000005845	GCF_000005845_030630	T4P_pilM	3063	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	3	0.556
→260	1	T4P_pilM	259	1.1e-09
→300	0.988	0.598	8	39.
GCF_000005845	GCF_000005845_026740	T4P_pilT	2674	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	-1	0.556
→260	1	T4P_pilT	326	1.1e-117
→600	0.944	0.979	3	393.
GCF_000005845	GCF_000005845_026930	T2SS_gspO	2693	TFF-SF/
→T4P	GCF_000005845_T4P_14	3	-2	0.556
→260	1	T4P_pilD	269	1.3e-87
→000	1.000	0.859	30	294.
→030080	GCF_000005845_T2SS_2		260	GCF_000005845_

Note

If a loner component is not clustered with other genes, it will not be considered as part of a locus. Thus, its locus number will be a negative value (numbered from -1) and will not be counted in the variable *sys_loci* (number of loci for a system). See above lines for more details.

GCF_000005845	GCF_000005845_026740	T4P_pilT	2674	TFF-SF/T4P	GCF_
→000005845_T4P_25	3	-1	0.556	7.800	
GCF_000005845	GCF_000005845_026930	T2SS_gspO	2693	TFF-SF/T4P	GCF_
→000005845_T4P_25	3	-2	0.556	7.800	

best_solution.tsv and all_best_solutions.tsv

Since MacSyFinder 2.0, a combinatorial exploration of solutions using sets of systems found is performed. We call best solution, the combination of systems offering the highest score.

The *best_solution.tsv* and *all_best_solutions.tsv* files have the same structure as the file *all_systems.tsv*, except that there is an extra column **sol_id** which is a solution identifier added to the file *all_best_solutions.tsv*. The systems that have the same “sol_id” belong to a same solution.

As the files have the same structure as *all_systems.tsv*, they can also be parsed with pandas as shown above.

For the description of the fields of *best_solution.tsv*, see [above](#) those of the *all_systems.tsv* file.

For the *all_best_solutions.tsv*, each line corresponds to a “hit” that has been assigned to a detected system. It includes:

- **sol_id** - the name of the solution it is part of (**only in *all_best_solutions.tsv* files**)
- **replicon** - the name of the replicon it belongs to
- **hit_id** - the unique identifier of the hit
- **gene_name** - the name of the component identified by the hit
- **hit_pos** - the position of the sequence in the replicon
- **model_fqn** - the model fully-qualified name
- **sys_id** - the unique identifier attributed to the detected system
- **sys_loci** - the number of loci
- **locus_num** - the number of the locus where is located this gene. Loners gene have negative locus_num
- **sys_wholeness** - the wholeness of the system
- **sys_score** - the system score
- **sys_occ** - the estimated number of system occurrences that could be potentially “filled” with this system’s occurrence, based on the average number of each component found. A proxy for the genetic potential ton encode several systems from the set of components found in this one occurrence.
- **hit_gene_ref** - the gene in the model whose this hit plays the role of
- **hit_status** - the status of the component in the assigned system’s definition
- **hit_seq_len** - the length of the protein sequence matched by this hit
- **hit_i_eval** - Hmmer statistics, the independent-evalue
- **hit_score** - Hmmer score
- **hit_profile_cov** - the percentage of the profile covered by the alignment with the sequence
- **hit_seq_cov** - the percentage of the sequence covered by the alignment with the profile
- **hit_begin_match** - the position in the sequence where the profile match begins
- **hit_end_match** - the position in the sequence where the profile match ends

- **counterpart** - the hit id of some other hit which are equivalent. Only loners and multi-systems hits have counterparts
- **used_in** - whether the hit could be used in another system's occurrence

Note

Each system reported is separated from the others with a blank line to ease human reading. These lines are ignored during the parsing with pandas.

Example of *best_solution.tsv* files

```
# macsyfinder 20220121.dev
# models : functional-0.0b2
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsyfinder --db-
→type=gembase --models-dir=tests/data/models/ --models TFF-SF Archaeal-T4P ComM MSH_
→T2SS T4bP T4P Tad --relative-path --sequence-db tests/data/base/gembase.fasta -w 12
# Systems found:
replicon      hit_id      gene_name      hit_pos      model_fqn      sys_
→id      sys_loci      locus_num      sys_wholeness      sys_score      sys_
→occ      hit_gene_ref      hit_status      hit_seq_len      hit_i_
→eval      hit_score      hit_profile_cov      hit_seq_cov      hit_begin_
→match      hit_end_match      counterpart      used_in
GCF_000005845      GCF_000005845_000970      T4P_pilC      97      TFF-SF/
→T4P      GCF_000005845_T4P_9      1      1      0.278      3.
→760      1      T4P_pilC      mandatory      400      2.2e-105      353.
→100      0.991      0.830      62      393
GCF_000005845      GCF_000005845_000980      T4P_pilB      98      TFF-SF/
→T4P      GCF_000005845_T4P_9      1      1      0.278      3.
→760      1      T4P_pilB      mandatory      461      8.9e-152      506.
→100      0.948      0.850      62      453
GCF_000005845      GCF_000005845_000990      T4P_pilA      99      TFF-SF/
→T4P      GCF_000005845_T4P_9      1      1      0.278      3.
→760      1      T4P_pilA      accessory      146      1.1e-19      71.
→200      0.859      0.473      5      73
GCF_000005845      GCF_000005845_026740      T4P_pilT      2674      TFF-SF/
→T4P      GCF_000005845_T4P_9      1      -1      0.278      3.
→760      1      T4P_pilT      mandatory      326      1.1e-117      393.
→600      0.944      0.979      3      321
GCF_000005845      GCF_000005845_026930      T2SS_gsp0      2693      TFF-SF/
→T4P      GCF_000005845_T4P_9      1      -2      0.278      3.
→760      1      T4P_pilD      mandatory      269      1.3e-87      294.
→000      1.000      0.859      30      260      GCF_000005845_
→030080      GCF_000005845_T2SS_2

GCF_000005845      GCF_000005845_025680      T4P_pilW      2568      TFF-SF/
→T4P      GCF_000005845_T4P_13      2      1      0.389      4.
→760      1      T4P_pilW      accessory      187      3.3e-08      34.
→500      0.625      0.401      6      80
GCF_000005845      GCF_000005845_025690      T4P_fimT      2569      TFF-SF/
→T4P      GCF_000005845_T4P_13      2      1      0.389      4.
→760      1      T4P_fimT      accessory      156      2.5e-06      28.
→500      0.939      0.397      5      66
```

(continues on next page)

(continued from previous page)

GCF_000005845	GCF_000005845_030590	T4P_pilQ	3059	TFF-SF/		
↪T4P	GCF_000005845_T4P_13	2	2	0.389	4.	
↪760	1	T4P_pilQ	mandatory	412	5.9e-51	173.
↪100	0.919	0.408	244	411		
GCF_000005845	GCF_000005845_030620	T4P_pilN	3062	TFF-SF/		
↪T4P	GCF_000005845_T4P_13	2	2	0.389	4.	
↪760	1	T4P_pilN	mandatory	179	3.8e-09	37.
↪500	0.986	0.765	5	141		

Example of *all_best_solutions.tsv* files

```
# macsyfinder 20220121.dev
# models : functional-0.0b2
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsfinder --db-
→type=gembase --models-dir=tests/data//models/ --models TFF-SF Archaeal-T4P ComM MSH_
→T2SS T4bP T4P Tad --relative-path --sequence-db tests/data/base/gembase.fasta -w 12
# Systems found:
sol_id      replicon      hit_id      gene_name      hit_pos      model_
→fqn        sys_id        sys_loci    locus_num      sys_wholessness  sys_
→score      sys_occ      hit_gene_ref hit_status      hit_seq_
→len        hit_i_eval    hit_score    hit_profile_cov hit_seq_
→cov        hit_begin_match hit_end_match counterpart      used_in
1          GCF_000005845 GCF_000005845_000970 T4P_pilC      97          TFF-
→SF/T4P      GCF_000005845_T4P_9 1          1          0.278      3.
→760        1          T4P_pilC      mandatory      400         2.2e-105    353.
→100        0.991      0.830        62          393
1          GCF_000005845 GCF_000005845_000980 T4P_pilB      98          TFF-
→SF/T4P      GCF_000005845_T4P_9 1          1          0.278      3.
→760        1          T4P_pilB      mandatory      461         8.9e-152    506.
→100        0.948      0.850        62          453
1          GCF_000005845 GCF_000005845_000990 T4P_pilA      99          TFF-
→SF/T4P      GCF_000005845_T4P_9 1          1          0.278      3.
→760        1          T4P_pilA      accessory      146         1.1e-19     71.
→200        0.859      0.473        5          73
1          GCF_000005845 GCF_000005845_026740 T4P_
→pilT        2674        TFF-SF/T4P      GCF_000005845_T4P_9 1          -
→1          0.278      3.760        1          T4P_
→pilT        mandatory      326          1.1e-117    393.600    0.944      0.
→979        3          321
1          GCF_000005845 GCF_000005845_026930 T2SS_
→gsp0        2693        TFF-SF/T4P      GCF_000005845_T4P_9 1          -
→2          0.278      3.760        1          T4P_
→pilD        mandatory      269          1.3e-87     294.000    1.000      0.
→859        30         260          GCF_000005845_030080 GCF_000005845_T2SS_2
1          GCF_000005845 GCF_000005845_025680 T4P_
→pilW        2568        TFF-SF/T4P      GCF_000005845_T4P_
→13         2          1          0.389      4.760      1          T4P_
→pilW        accessory      187          3.3e-08     34.500     0.625      0.
→401        6          80
1          GCF_000005845 GCF_000005845_025690 T4P_
→fimT        2569        TFF-SF/T4P      GCF_000005845_T4P_
```

(continues on next page)

(continued from previous page)

→13	2	1	0.389	4.760	1	T4P_		
→fimT		accessory	156	2.5e-06	28.500		0.939	0.
→397	5	66						
1	GCF_000005845	GCF_000005845_030590				T4P_		
→pilQ	3059	TFF-SF/T4P				GCF_000005845_T4P_		
→13	2	2	0.389	4.760	1	T4P_		
→pilQ		mandatory	412	5.9e-51	173.100		0.919	0.
→408	244	411						
1	GCF_000005845	GCF_000005845_030620				T4P_		
→pilN	3062	TFF-SF/T4P				GCF_000005845_T4P_		
→13	2	2	0.389	4.760	1	T4P_		
→pilN		mandatory	179	3.8e-09	37.500		0.986	0.
→765	5	141						
1	GCF_000005845	GCF_000005845_030630				T4P_		
→pilM	3063	TFF-SF/T4P				GCF_000005845_T4P_		
→13	2	2	0.389	4.760	1	T4P_		
→pilM		accessory	259	1.1e-09	39.300		0.988	0.
→598	8	162						
1	GCF_000005845	GCF_000005845_026740				T4P_		
→pilT	2674	TFF-SF/T4P				GCF_000005845_T4P_13	2	-
→1	0.389	4.760	1			T4P_		
→pilT		mandatory	326	1.1e-117	393.600		0.944	0.
→979	3	321						
1	GCF_000005845	GCF_000005845_026930				T2SS_		
→gsp0	2693	TFF-SF/T4P				GCF_000005845_T4P_13	2	-
→2	0.389	4.760	1			T4P_		
→pilD		mandatory	269	1.3e-87	294.000		1.000	0.
→859	30	260				GCF_000005845_030080	GCF_000005845_T2SS_2	
1	GCF_000005845	GCF_000005845_029970				T2SS_		
→gspC	2997	TFF-SF/T2SS				GCF_000005845_T2SS_		
→1	1	1	0.857	9.000	1	T2SS_		
→gspC		mandatory	271	2.3e-19	70.400		0.897	0.
→358	47	143						
1	GCF_000005845	GCF_000005845_030050				T2SS_		
→gspK	3005	TFF-SF/T2SS				GCF_000005845_T2SS_		
→1	1	1	0.857	9.000	1	T2SS_		
→gspK		accessory	327	1e-16	61.500		1.000	0.
→180	6	64						
1	GCF_000005845	GCF_000005845_030060				T2SS_		
→gspL	3006	TFF-SF/T2SS				GCF_000005845_T2SS_		
→1	1	1	0.857	9.000	1	T2SS_		
→gspL		accessory	387	1.5e-37	129.300		1.000	0.
→351	6	141						
1	GCF_000005845	GCF_000005845_030070				T2SS_		
→gspM	3007	TFF-SF/T2SS				GCF_000005845_T2SS_		
→1	1	1	0.857	9.000	1	T2SS_		
→gspM		accessory	153	2.8e-29	102.900		0.985	0.
→804	13	135						
1	GCF_000005845	GCF_000005845_030080				T2SS_		
→gsp0	3008	TFF-SF/T2SS				GCF_000005845_T2SS_		
→1	1	1	0.857	9.000	1	T2SS_		

(continues on next page)

(continued from previous page)

```

→gsp0          mandatory          225          4e-65          220.400          0.978          0.
→840           26           214

# WARNING Loner: there is only 1 occurrence(s) of loner 'T4P_pilT' and 2 potential_
→systems [GCF_000005845_T4P_9, GCF_000005845_T4P_13]

2          GCF_000005845          GCF_000005845_000970          T4P_pilC          97          TFF-
→SF/T4P          GCF_000005845_T4P_11          2          1          0.389          4.
→760           1          T4P_pilC          mandatory          400          2.2e-105          353.
→100           0.991          0.830          62          393

```

Note

If a loner component is not clustered with other genes, it will not be considered as part of a locus. Thus, its locus number will be a negative value (numbered from -1) and will not be counted in the variable *sys_loci* (number of loci for a system). See above lines for more details.

Note

If several systems from same model use a loner (same gene) *msf* check that there is at least one occurrence of this hit for each system. If there are fewer hits than systems occurrence a warning is displayed in *best_solution.tsv* or *all_best_solution.tsv* as comment. So the file can be parsed with pandas without problem.

```

1          GCF_000005845          GCF_000005845_030080          T2SS_gsp0          3008          TFF-SF/T2SS
→GCF_000005845_T2SS_1          1          1          0.857          9.000          1          T2SS_gsp0
→mandatory          225          4e-65          220.400          0.978          0.840          26          214

# WARNING Loner: there is only 1 occurrence(s) of loner 'T4P_pilT' and 2 potential_
→systems [GCF_000005845_T4P_9, GCF_000005845_T4P_13]

2          GCF_000005845          GCF_000005845_000970          T4P_pilC          97          TFF-SF/T4P
→GCF_000005845_T4P_11          2          1          0.389          4.760          1          T4P_pilC
→mandatory          400          2.2e-105          353.100          0.991          0.830          62          393

```

Note

In case multiple solutions have the exact same score, a sorting is performed among the best solutions, and the solution ranked 1st is reported in the *best_solution.tsv* and *best_solution.txt* files. The ranking is performed as follow:

1. by the number of systems' components (hits) constituting the solution (most components first)
2. by the number of systems (most systems in first)
3. by the average of systems' wholeness
4. by hits position. This criterion is mostly introduced to produce reproducible results between two runs.

best_solution_summary.tsv

This file is a concise view of which systems have been found in your replicons and how many per replicon. It is based on **best_solution.tsv**. The first two lines are comments that indicate the version of MacSyFinder and the command line used to generate the results. Then a table represented by tabulated text to separate columns, with the searched models in columns and the replicons scanned for the models in row.

```
# macsyfinder 20220121.dev
# models : functional-0.0b2
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsyfinder --db-
→type=gembase --models-dir=tests/data//models/ --models TFF-SF Archaeal-T4P ComM MSH_
→T2SS T4bP T4P Tad --relative-path --sequence-db tests/data/base/gembase.fasta -w 12
```

replicon	TFF-SF/MSH	TFF-SF/T2SS	TFF-SF/T4P	TFF-SF/ComM	TFF-SF/T4bP	TFF-SF/Tad	TFF-SF/Archaeal-T4P
GCF_0000005845	0	1	2	0	0	0	0
GCF_0000006725	0	1	2	0	0	0	0
GCF_0000006745	1	1	2	1	0	0	0
GCF_0000006765	0	3	1	0	1	0	0
GCF_0000006845	0	0	1	0	0	0	0
GCF_0000006905	0	1	0	0	1	0	0
GCF_0000006925	0	0	1	0	0	0	0
GCF_0000006945	0	0	2	0	0	0	0

as a *tsv* file it can be parsed easily using pandas:

```
import pandas as pd
solution = pd.read_csv('path to best_solution_summary.tsv', sep='\t', comment='#', index_
→col=0)
```

Note

If you want to do the same operation but based on the *all_best_solutions.tsv* file, you can do it with the few lines of pandas below:

```
import pandas as pd

all_best_sol = '<macsyfinder_results_dir>/all_best_solutions.tsv'

# read data from best_solution file
data = pd.read_csv(all_best_sol, sep='\t', comment='#')

# remove useless columns
selection = data[['sol_id', 'replicon', 'sys_id', 'model_fqn']]

# keep only one row per replicon, sys_id
dropped = selection.drop_duplicates(subset=['sol_id', 'replicon', 'sys_id'])

# count for each replicon which models have been detected and their
→occurrences
summary = pd.crosstab(index=[dropped.sol_id, dropped.replicon],
→columns=dropped['model_fqn'])
```

if you are not fluent in *pandas*, we provide you a tiny script *msf_summary.py* based on few lines above to do the job *msf_summary.py*.

Then you can run the script

```
python msf_summary.py <path_to_all_best_solutions.tsv>
```

below an example of summary of *all_best_solutions.tsv*

sol_id	replicon	TFF-SF/MSH		TFF-SF/T2SS		TFF-SF/
↪ T4P	TFF-SF/T4bP	TFF-SF/Tad				
1	GCF_0000005845	0	1	1	0	0
2	GCF_0000006725	0	1	1	0	0
3	GCF_0000006725	0	1	1	0	0
4	GCF_0000006745	1	1	2	1	0
5	GCF_0000006745	1	1	2	1	0
6	GCF_0000006745	1	1	1	1	0
7	GCF_0000006765	0	3	1	0	1
8	GCF_0000006845	0	0	1	0	0
9	GCF_0000006905	0	1	0	0	1
10	GCF_0000006925	0	0	1	0	0
11	GCF_0000006945	0	0	1	0	0

best_solution_loners.tsv

This file give an overview of all hits identified as Loner in the best_solution

```
# macsyfinder 20220121.dev
# models : functional-0.0b2
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsfinder --db-
↪ type=gembase --models-dir=tests/data//models/ --models TFF-SF Archaeal-T4P_
↪ ComM MSH T2SS T4bP T4P Tad --relative-path --sequence-db tests/data/base/
↪ gembase.fasta -w 12
# Loners found:
replicon      model_fqn      function      gene_name      hit_
↪ id          hit_pos      hit_status      hit_seq_len      hit_i_
↪ eval        hit_score      hit_profile_cov      hit_seq_cov      hit_
↪ begin_match      hit_end_match
GCF_0000005845      TFF-SF/T4P      T4P_pilT      T4P_pilT      GCF_
↪ 0000005845_026740      2674      mandatory      326      1.100e-
↪ 117      393.600      0.944      0.979      3      321
GCF_0000005845      TFF-SF/T4P      T4P_pilD      T2SS_gsp0      GCF_
↪ 0000005845_026930      2693      mandatory      269      1.300e-
↪ 87      294.000      1.000      0.859      30      260
GCF_0000005845      TFF-SF/T4P      T4P_pilD      T2SS_gsp0      GCF_
↪ 0000005845_030080      3008      mandatory      225      4.000e-
↪ 65      220.400      0.978      0.840      26      214
GCF_0000006725      TFF-SF/T4P      T4P_pilT      T4P_pilT      GCF_
↪ 0000006725_000270      4269      mandatory      344      1.800e-
↪ 172      573.700      0.994      0.985      2      340
GCF_0000006725      TFF-SF/T4P      T4P_pilA      T4P_pilA      GCF_
↪ 0000006725_003680      4610      accessory      187      9.000e-
↪ 10      39.500      0.667      0.278      6      57
GCF_0000006725      TFF-SF/T2SS      T2SS_gsp0      T4P_pilD      GCF_
↪ 0000006725_014570      5699      mandatory      287      7.400e-
↪ 77      258.600      1.000      0.836      28      267
```

(continues on next page)

(continued from previous page)

GCF_000006725	TFF-SF/T2SS	T2SS_gspE	T2SS_gspE	GCF_	
↪000006725_018700	6112	mandatory	566	1.800e-	
↪171	571.000	0.936	0.701	165	561
GCF_000006725	TFF-SF/T4P	T4P_pilA	T4P_pilA	GCF_	
↪000006725_022640	6506	accessory	178	2.000e-	
↪10	41.600	0.603	0.264	5	51
GCF_000006745	TFF-SF/T2SS	T2SS_gsp0	T4P_pilD	GCF_	
↪000006745_021980	8766	mandatory	291	3.100e-	
↪88	295.800	1.000	0.832	28	269
GCF_000006765	TFF-SF/T2SS	T2SS_gsp0	T4P_pilD	GCF_	
↪000006765_044730	14545	mandatory	290	1.100e-	
↪88	297.200	1.000	0.828	31	270
GCF_000006925	TFF-SF/T4P	T4P_pilT	T4P_pilT	GCF_	
↪000006925_026070	23874	mandatory	341	6.600e-	
↪118	394.300	0.950	0.941	18	338
GCF_000006945	TFF-SF/T4P	T4P_pilT	T4P_pilT	GCF_	
↪000006945_030160	28596	mandatory	326	3.400e-	
↪113	378.800	0.933	0.966	3	317
GCF_000006945	TFF-SF/T4P	T4P_pilD	T2SS_gsp0	GCF_	
↪000006945_033450	28925	mandatory	155	2.900e-	
↪35	122.700	0.588	0.871	9	143

best_solution_multisystems.tsv

This file give an overview of all hits identified as multi-systems in the best_solution

```
# macsyfinder 20220121.dev
# models : functional-0.0b2
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsfinder --db-
→type ordered_replicon --replicon-topology linear --models-dir tests/data/
→models/ -m functional T12SS-multisystem --relative-path --sequence-db tests/
→data/base/test_13.fasta -w 15
# Multisystems found:
replicon      model_fqn      function      gene_name      hit_
→id      hit_pos      hit_status      hit_seq_len      hit_i_
→eval      hit_score      hit_profile_cov      hit_seq_cov      hit_
→begin_match      hit_end_match
UserReplicon      functional/T12SS-multisystem      T1SS_omf      T1SS_
→omf      VICH001.B.00001.C001_
→01360      20      mandatory      484      3.200e-28      90.
→000      0.985      0.820      80      476
UserReplicon      functional/T12SS-multisystem      T1SS_omf      T1SS_
→omf      VICH001.B.00001.C001_
→01506      35      mandatory      419      9.100e-35      111.
→500      0.998      0.912      25      406
```

rejected_candidates.txt

This file records all clusters or cluster combinations (if the “multi_loci” search mode is on) which have been discarded and the reason why they were not selected as systems.

The header is composed of the MacSyFinder version and the command line used followed by the description of the cluster(s). The list of the hits composing the cluster is presented at the end of the cluster or clusters’ combination,

followed by the reason why it has been discarded.

Note

This file is in human readable format. If you need to parse the information about rejected candidates, use the tsv formatted file `rejected_candidates.tsv`

```
# macsyfinder 20200511.dev
# models : TFF-SF-0.1b
# macsyfinder --sequence-db data/base/GCF_0000006745.fasta --models TFF-SF all --models-
↳dir data/models/ --db-type gembase -w 4
# Rejected candidates:

Cluster:
- model: T4P
- hits: (GCF_0000005845_025680, T4P_pilW, 2568), (GCF_0000005845_025690, T4P_fimT,
↳2569)
Cluster:
- model: T4P
- hits: (GCF_0000005845_026930, T2SS_gsp0, 2693)
Cluster:
- model: T4P
- hits: (GCF_0000005845_030080, T2SS_gsp0, 3008)
This candidate has been rejected because:
The quorum of mandatory genes required (4) is not reached: 1
The quorum of genes required (5) is not reached: 3
=====
Cluster:
- model: Archaeal-T4P
- hits: (GCF_0000005845_019260, Archaeal-T4P_arCOG00589, 1926), (GCF_0000005845_019310,
↳ Archaeal-T4P_arCOG02900, 1931)
This candidate has been rejected because:
The quorum of mandatory genes required (3) is not reached: 0
The quorum of genes required (3) is not reached: 2
=====
```

rejected_candidates.tsv

This file contains same information as *rejected_candidates.txt* but in tsv format, so it's more convenient to parse it. for instance with python and [pandas](#) library.:

```
import pandas as pd
pd.read_csv("path/to/rejected_candidates.tsv", sep='\t', comment='#')
```

As other file the first lines are comments and provides informations to indicate how this file has been produced.

- the macsyfinder version
- the model package and version used
- the command line used

then the following information separated by 'tabulation' character 't'

- **candidate_id** - An unique identifier of the candidate (for this run)

- **replicon** - The name of the replicon
- **model_fqn** - The model fully-qualified name
- **cluster_id** - An unique identifier for the cluster constituting the candidate
- **hit_id** - The identifier of the hit (as indicate in hmmer output)
- **hit_pos** - The position of the sequence in the replicon
- **gene_name** - The name of the component identified by the hit
- **function** - The name of the gene for which it it fulfill the function.
- **reasons** - The reasons why this cluster has been discarded. ther can be several reasons, in this case each reason are separated by '/

Note

A rejected candidate can be constituted of

- clusters (can have several clusters if the model is multi loci),
- loners

Example of *rejected_candidates.tsv*

```
# macsyfinder 20220805.dev
# models : TFF-SF-None
# /home/bneron/Projects/GEM/MacSyFinder/MacSyFinder/py39/bin/macsyfinder --sequence-db_
↳data/base/GCF_000006745.fasta --models TFF-SF all --models-dir data/models/ --db-type_
↳gembase -w 15
# Rejected candidates found:
candidate_id      replicon      model_fqn      cluster_id      hit_id      hit_
↳pos      gene_name      function      reasons
GCF_000006745_Archaeal-T4P_1      GCF_000006745      TFF-SF/Archaeal-
↳T4P      c3      GCF_000006745_018740      1874      Archaeal-T4P_
↳arCOG00589      Archaeal-T4P_arCOG00589      The quorum of mandatory genes_
↳required (3) is not reached: 0/The quorum of genes required (3) is not reached: 1
GCF_000006745_Archaeal-T4P_1      GCF_000006745      TFF-SF/Archaeal-
↳T4P      c3      GCF_000006745_018800      1880      Archaeal-T4P_
↳arCOG00589      Archaeal-T4P_arCOG00589      The quorum of mandatory genes_
↳required (3) is not reached: 0/The quorum of genes required (3) is not reached: 1

GCF_000006745_Archaeal-T4P_2      GCF_000006745      TFF-SF/Archaeal-
↳T4P      c4      GCF_000006745_026670      2667      Archaeal-T4P_
↳arCOG02900      Archaeal-T4P_arCOG02900      The quorum of mandatory genes_
↳required (3) is not reached: 0/The quorum of genes required (3) is not reached: 1
GCF_000006745_Archaeal-T4P_2      GCF_000006745      TFF-SF/Archaeal-
↳T4P      c4      GCF_000006745_026680      2668      Archaeal-T4P_
↳arCOG02900      Archaeal-T4P_arCOG02900      The quorum of mandatory genes_
↳required (3) is not reached: 0/The quorum of genes required (3) is not reached: 1

GCF_000006745_ComM_4      GCF_000006745      TFF-SF/ComM      c11      GCF_
↳000006745_017080      1708      ComM_comEC      ComM_comEC      The quorum of_
↳mandatory genes required (4) is not reached: 1/The quorum of genes required (4) is not_
↳reached: 1
```

(continues on next page)

(continued from previous page)

GCF_000006745_ComM_5	GCF_000006745	TFF-SF/ComM	c12	GCF_
→000006745_032430	3243	ComM_comEB	ComM_comEB	The quorum of_
→mandatory genes required (4) is not reached: 1/The quorum of genes required (4) is not_				
→reached: 2				
GCF_000006745_ComM_5	GCF_000006745	TFF-SF/ComM	c13	GCF_
→000006745_017080	1708	ComM_comEC	ComM_comEC	The quorum of_
→mandatory genes required (4) is not reached: 1/The quorum of genes required (4) is not_				
→reached: 2				
GCF_000006745_ComM_3	GCF_000006745	TFF-SF/ComM	c10	GCF_
→000006745_032430	3243	ComM_comEB	ComM_comEB	The quorum of_
→mandatory genes required (4) is not reached: 0/The quorum of genes required (4) is not_				
→reached: 1				
GCF_000006745_MSH_6	GCF_000006745	TFF-SF/MSH	c18	GCF_
→000006745_004600	460	MSH_mshA	MSH_mshA	The quorum of_
→mandatory genes required (3) is not reached: 1/The quorum of genes required (4) is not_				
→reached: 1				
GCF_000006745_T2SS_7	GCF_000006745	TFF-SF/T2SS	c25	GCF_
→000006745_021980	2198	T4P_pilD	T2SS_gsp0	The quorum of_
→mandatory genes required (4) is not reached: 1/The quorum of genes required (6) is not_				
→reached: 1				
GCF_000006745_T4P_8	GCF_000006745	TFF-SF/T4P	c30	GCF_
→000006745_004240	424	T4P_pilT	T4P_pilT	The quorum of_
→mandatory genes required (4) is not reached: 1/The quorum of genes required (5) is not_				
→reached: 2				
GCF_000006745_T4P_8	GCF_000006745	TFF-SF/T4P	c30	GCF_
→000006745_004250	425	T4P_pilU	T4P_pilU	The quorum of_
→mandatory genes required (4) is not reached: 1/The quorum of genes required (5) is not_				
→reached: 2				
GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c34	GCF_
→000006745_004240	424	T4P_pilT	T4P_pilT	The quorum of_
→mandatory genes required (4) is not reached: 2				
GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c34	GCF_
→000006745_004250	425	T4P_pilU	T4P_pilU	The quorum of_
→mandatory genes required (4) is not reached: 2				
GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c35	GCF_
→000006745_007820	782	T4P_pilE	T4P_pilE	The quorum of_
→mandatory genes required (4) is not reached: 2				
GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c35	GCF_
→000006745_007830	783	T4P_fimT	T4P_fimT	The quorum of_
→mandatory genes required (4) is not reached: 2				
GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c35	GCF_
→000006745_007840	784	T4P_pilW	T4P_pilW	The quorum of_
→mandatory genes required (4) is not reached: 2				
GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c35	GCF_
→000006745_007850	785	T4P_pilX	T4P_pilX	The quorum of_
→mandatory genes required (4) is not reached: 2				

(continues on next page)

(continued from previous page)

GCF_000006745_T4P_12	GCF_000006745	TFF-SF/T4P	c35	GCF_
↪000006745_007860	786	T4P_pilV		The quorum of_
↪mandatory genes required (4) is not reached: 2				

Note

If a timeout is set to limit the time spent in best solution resolution. This timeout is applied per replicon. If the best solution resolution reach the timeout for a replicon, a WARNING is raised in *macsyfinder.log*. The warning is also report in the following files:

- *best_solution.tsv*
- *best_solution_summary.tsv*
- *all_best_solutions.tsv*
- *all_systems.tsv* and *all_systems.txt*
- *rejected_candidates.tsv* and *rejected_candidates.txt*

for instance

```
# macsyfinder 20230113.dev
# models : TXSScan-1.1.1
# macsyfinder --sequence-db tests/data/base/gembase.fasta --db-type gembase --models_
↪TXSScan -w 15 --timeout 1s
#
# WARNING: The replicon 'GCF_000006765' has been SKIPPED. Cannot be solved before_
↪timeout.
#
replicon hit_id ...
```

Output files for the “unordered replicon” search mode

Systems detection results

As for ordered replicons, several output files are provided.

- *all_systems.txt* - This file contains the description of candidate systems found.
- *all_systems.tsv* - The same information as in *all_systems.txt* but in the tabulated tsv format.
- *uncomplete_systems.txt* - This file contains occurrences for systems that did not complete models’ definitions and that were therefore not kept as candidate systems.

Note

In this *unordered* search mode, there is no notion of order or distance of the components along the replicon. The clustering step is skipped by MacSyFinder, and it is therefore “only” checked for each type of system being searched whether there is the genetic potential to fulfil its model definition.

all_systems.txt

This file contains potential systems for unordered replicon in human readable format.

In this file, for each component of each searched system's model, we report the number of hits found. For the description of the fields, see [above](#).

Warning

In this mode the *forbidden* genes are reported here to the user. As we do not know if they co-localize (cluster) with the other genes they could be present in the replicon, yet far away - or very close on the contrary - to the potential system.

```
# macsyfinder 20201028.dev
# models : TFF-SF-0.1b
# macsyfinder --sequence-db tests/data/base/one_replicon.fasta --db-type unordered --
↳models-dir tests/data/models -m TFF-SF T4P_single_locus
# Systems found:
```

This replicon contains genetic materials needed for system TFF-SF/T4P_single_locus

```
system id = Unordered_T4P_single_locus_1
model = TFF-SF/T4P_single_locus
replicon = Unordered
hits = [('GCF_000006845_000250', 'T4P_pilY', 25), ('GCF_000006845_000700', 'T4P_pilY',
↳70), ('GCF_000006845_001030', 'T4P_pilQ', 103), ('GCF_000006845_001040', 'T4P_pilP',
↳104), ('GCF_000006845_001050', 'T4P_pilO', 105), ('GCF_000006845_001060', 'T4P_pilN',
↳106), ('GCF_000006845_001070', 'T4P_pilM', 107), ('GCF_000006845_003200', 'T4P_pilU',
↳320), ('GCF_000006845_004190', 'T4P_fimT', 419), ('GCF_000006845_004200', 'T4P_pilV',
↳420), ('GCF_000006845_004210', 'T4P_pilW', 421), ('GCF_000006845_004220', 'T4P_pilX',
↳422), ('GCF_000006845_004230', 'T4P_pilA', 423), ('GCF_000006845_010160', 'T4P_pilA',
↳1016), ('GCF_000006845_012440', 'T4P_pilA', 1244), ('GCF_000006845_014270', 'T4P_pilC',
↳1427), ('GCF_000006845_014280', 'T4P_pilD', 1428), ('GCF_000006845_014310', 'T4P_pilB',
↳1431), ('GCF_000006845_016430', 'T4P_pilT', 1643), ('GCF_000006845_016440', 'T4P_
↳pilU', 1644)]
wholeness = 0.889
```

mandatory genes:

- T4P_pilE: 0 ()
- T4P_pilB: 1 (T4P_pilB)
- T4P_pilC: 1 (T4P_pilC)
- T4P_pilO: 1 (T4P_pilO)
- T4P_pilQ: 1 (T4P_pilQ)
- T4P_pilN: 1 (T4P_pilN)
- T4P_pilT: 1 (T4P_pilT)
- T4P_pilD: 1 (T4P_pilD)

accessory genes:

- T4P_pilA: 3 (T4P_pilA, T4P_pilA, T4P_pilA)
- T4P_pilV: 1 (T4P_pilV)
- T4P_pilY: 2 (T4P_pilY, T4P_pilY)

(continues on next page)

(continued from previous page)

```

- T4P_pilW: 1 (T4P_pilW)
- T4P_pilX: 1 (T4P_pilX)
- T4P_fimT: 1 (T4P_fimT)
- T4P_pilM: 1 (T4P_pilM)
- T4P_pilP: 1 (T4P_pilP)
- T4P_pilU: 2 (T4P_pilU, T4P_pilU)
- MSH_mshM: 0 ()

```

neutral genes:

forbidden genes:

Use ordered replicon to have better prediction.

all_systems.tsv

This file contains the same information as in *all_systems.txt* but in *tsv* format. For the description of the fields, see *above*.

Note

This file can be easily parsed with pandas:

```

import pandas as pd
pot_systems = pd.read_csv('all_systems.tsv', sep='\t', comment='#')

```

```

# macsyfinder 20201028.dev
# models : TFF-SF-0.1b
# macsyfinder --sequence-db tests/data/base/one_replicon.fasta --db-type unordered --
↳models-dir tests/data/models -m TFF-SF T4P_single_locus
# Likely Systems found:

replicon   hit_id  gene_name      hit_pos model_fqn      sys_id sys_wholeness hit_
↳gene_ref  hit_status hit_seq_len hit_i_eval hit_score hit_
↳profile_cov hit_seq_cov hit_begin_match hit_end_match used_in
Unordered  GCF_0000006845_014310 T4P_pilB      1431 TFF-SF/T4P_single_locus_
↳Unordered_T4P_single_locus_1 0.889 T4P_pilB      mandatory 558 3.8e-
↳178 589.000 0.964 0.731 146 553
Unordered  GCF_0000006845_014270 T4P_pilC      1427 TFF-SF/T4P_single_locus_
↳Unordered_T4P_single_locus_1 0.889 T4P_pilC      mandatory 410 1.9e-
↳131 434.800 0.997 0.817 72 406
Unordered  GCF_0000006845_014280 T4P_pilD      1428 TFF-SF/T4P_single_locus_
↳Unordered_T4P_single_locus_1 0.889 T4P_pilD      mandatory 286 2.8e-
↳82 272.300 1.000 0.829 28 264
Unordered  GCF_0000006845_001060 T4P_pilN      106 TFF-SF/T4P_single_locus_
↳Unordered_T4P_single_locus_1 0.889 T4P_pilN      mandatory 199 2.3e-
↳33 112.200 0.986 0.714 7 148
Unordered  GCF_0000006845_001050 T4P_pilO      105 TFF-SF/T4P_single_locus_
↳Unordered_T4P_single_locus_1 0.889 T4P_pilO      mandatory 215 2.9e-
↳37 124.800 0.980 0.693 23 171

```

(continues on next page)

(continued from previous page)

Unordered	GCF_000006845_001030	T4P_pilQ	103	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilQ		mandatory 723 1.9e-
↳ 62 206.600 0.935 0.238 548		719		
Unordered	GCF_000006845_016430	T4P_pilT	1643	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilT		mandatory 347 6.9e-
↳ 167 551.400 0.997 0.983 2 342				
Unordered	GCF_000006845_004190	T4P_fimT	419	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_fimT		accessory 221 2.7e-
↳ 23 78.900 0.985 0.294 7 71				
Unordered	GCF_000006845_004230	T4P_pilA	423	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilA		accessory 162 8.6e-
↳ 20 67.800 0.744 0.389 9 71				
Unordered	GCF_000006845_010160	T4P_pilA	1016	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilA		accessory 149 1.3e-
↳ 15 54.300 0.821 0.430 5 68				
Unordered	GCF_000006845_012440	T4P_pilA	1244	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilA		accessory 129 1.5e-
↳ 19 67.000 0.859 0.519 6 72				
Unordered	GCF_000006845_001070	T4P_pilM	107	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilM		accessory 371 3.3e-
↳ 43 144.300 0.988 0.429 30 188				
Unordered	GCF_000006845_001040	T4P_pilP	104	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilP		accessory 181 2.7e-
↳ 34 115.600 1.000 0.735 13 145				
Unordered	GCF_000006845_003200	T4P_pilU	320	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilU		accessory 376 2.2e-
↳ 170 562.600 0.985 0.896 16 352				
Unordered	GCF_000006845_016440	T4P_pilU	1644	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilU		accessory 408 1.5e-
↳ 127 421.800 0.994 0.833 40 379				
Unordered	GCF_000006845_004200	T4P_pilV	420	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilV		accessory 203 9.6e-
↳ 16 54.600 1.000 0.276 14 69				
Unordered	GCF_000006845_004210	T4P_pilW	421	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilW		accessory 326 1.7e-
↳ 10 38.000 0.517 0.190 17 78				
Unordered	GCF_000006845_004220	T4P_pilX	422	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilX		accessory 203 2.8e-
↳ 18 62.600 0.983 0.286 17 74				
Unordered	GCF_000006845_000250	T4P_pilY	25	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilY		accessory 1006 2.2e-
↳ 57 191.700 0.728 0.389 463 853				
Unordered	GCF_000006845_000700	T4P_pilY	70	TFF-SF/T4P_single_locus_
↳ Unordered_T4P_single_locus_1	0.889	T4P_pilY		accessory 1047 1.9e-
↳ 57 191.900 0.721 0.362 516 894				

uncomplete_systems.txt

This file is created when a search is performed in the *unordered replicon* mode. This file list models that probably do not have not full systems in the replicon(s). For each model, the reason why it is not fulfilled is reported, followed by the model description and the components found.

```

# macsyfinder 20201113.dev
# models : TFF-SF-0.1b
# macsyfinder --sequence-db tests/data/base/one_replicon.fasta --db-type unordered --
↳models-dir tests/data/models -m TFF-SF all
# Unlikely Systems found:

This replicon probably not contains a system TFF-SF/T2SS:
The quorum of mandatory genes required (4) is not reached: 1
The quorum of genes required (6) is not reached: 2

system id = Unordered_T2SS_3
model = TFF-SF/T2SS
replicon = Unordered
hits = [('GCF_000006845_002600', 'Tad_tadD', 260), ('GCF_000006845_014280', 'T4P_pilD',
↳1428), ('GCF_000006845_016430', 'T4P_pilT', 1643)]
wholeness = 0.143

mandatory genes:
  - T2SS_gspD: 0 ()
  - T2SS_gspE: 0 ()
  - T2SS_gspF: 0 ()
  - T2SS_gspG: 0 ()
  - T2SS_gspC: 0 ()
  - T2SS_gspO: 1 (T4P_pilD)

accessory genes:
  - T2SS_gspM: 0 ()
  - T2SS_gspH: 0 ()
  - T2SS_gspI: 0 ()
  - T2SS_gspJ: 0 ()
  - T2SS_gspK: 0 ()
  - T2SS_gspN: 0 ()
  - T2SS_gspL: 0 ()
  - Tad_tadD: 1 (Tad_tadD)

neutral genes:

forbidden genes:
  - T4P_pilT: 1 (T4P_pilT)

Use ordered replicon to have better prediction.

=====

```

Hammer results' output files

Raw Hammer outputs are provided, as long with processed tabular outputs that include hits filtered as specified by the user. For instance, the Hammer search for SctC homologs with the corresponding profile will result in the creation of two output files: “sctC.search_hmm.out” for the raw HMMER output file and “sctC.res_hmm_extract” for the output file after processing/filtering of the HMMER results by MacSyFinder.

The processed output file “sctC.res_hmm_extract” recalls on the first lines the parameters used for hits filtering and relevant information on the matches, as in this example:

```
# gene: sctC extract from /Users/bob/macsyfinder_results/
      macsyfinder-20130128_08-57-46/sctC.search_hmm.out hmm output
# profile length= 544
# i_evalue threshold= 0.001000
# coverage threshold= 0.500000
# hit_id replicon_name position_hit hit_sequence_length gene_name gene_system i_evalue
→score
      profile_coverage sequence_coverage begin end
PSAE001c01_006940      PSAE001c01      3450      803      sctC      T3SS      1.1e-41 141.6
      0.588235 0.419676      395      731
PSAE001c01_018920      PSAE001c01      4634      776      sctC      T3SS      9.2e-48 161.7
      0.976103 0.724227      35      596
PSAE001c01_031420      PSAE001c01      5870      658      sctC      T3SS      2.7e-52 176.7
      0.963235 0.844985      49      604
PSAE001c01_051090      PSAE001c01      7801      714      sctC      T3SS      1.9e-46 157.4
      0.571691 0.463585      374      704
```

Logs and configuration files

Three specific output files are systematically built, whatever the search mode, to store information on MacSyFinder's execution:

- **macsyfinder.conf** - contains the configuration information of the run. It is useful to recover all the parameters used for the run.
- **macsyfinder.log** - the log file, contains raw information on the run. Please send it to us with any **bug report**.

For big data people

Parallelization

The time limiting part are HMMER (search genes). If you want to deal with a large data

- a collection of file containing replicons (each file must contains one replicon)
- or a *gembase* file (with a lot of replicons, from ten to more than thousand)

we provide a workflow to parallelize the execution by the data. This mean that

1. We split the data input into chunks containing one replicon each (for *gembase* input file).
2. Then execute MacSyFinder in parallel on each replicon (the number of parallel tasks can be limited)
3. Then aggregate the results in one global summary.

The workflow use the [nextflow](#) framework and can be run on a single machine or a cluster.

First, you have to install [nextflow](#) first, and [macsyfinder](#). Then we provide 2 files (you need to download them from the MacSyFinder github repo.)

- *parallel_macsyfinder.nf* which is the workflow itself in nextflow syntax
- *nextflow.config* which is a configuration file to execute the workflow.

The workflow file should not be modified. Whereas the profile **must** be adapted to the **local** architecture.

The file *nextflow.config* provide five profiles:

- a standard profile for local use (single machine).
- an apptainer profile using docker image with apptainer executor (on a single machine).

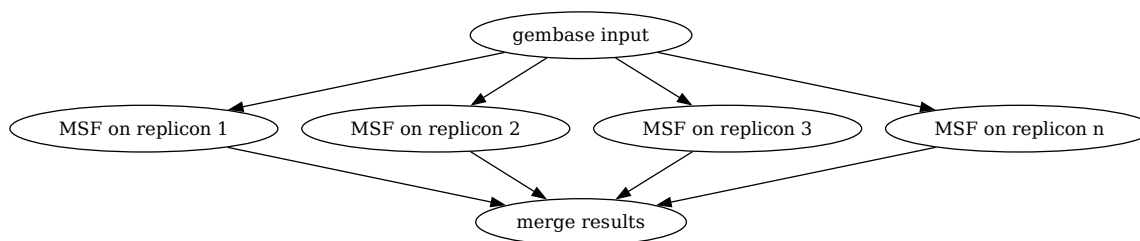


Fig. 1: Diagram of the parallel_macsyfinder workflow on gembase input

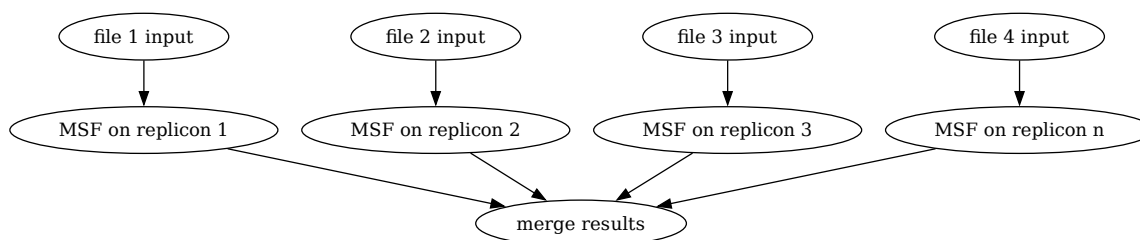


Fig. 2: Diagram of the parallel_macsyfinder workflow on ordered or unordered replicon

- a docker profile using docker image with docker executor (on a single machine).
- a cluster profile.
- a cluster profile using apptainer container system with the docker image.

How to get parallel_macsyfinder

The release contains the workflow *parallel_macsyfinder.nf* and the *nextflow.config* at the top level of the archive. But if you use pip to install MacSyFinder you have not easily access to them. But they can be downloaded or executed directly by using nextflow.

to download it

```
nextflow pull gem-pasteur/macsyfinder
```

to get the latest version or use *-r* option to specify a version

```
nextflow pull -r release_2.0 gem-pasteur/macsyfinder
```

to see what you download

```
nextflow view macsyfinder
```

to execute it directly on a local host with macsyfinder already installed and with models installed too:

```
nextflow run gem-pasteur/macsyfinder -profile standard --models "TFF-SF all" --db-type_
→gembase --sequence-db <path/to/my/gembase.fasta>
```

or:

```
nextflow run -r release_2.0 gem-pasteur/macsyfinder -profile standard --models "TFF-SF_
↳all" --db-type gembas --sequence-db <path/to/my/gembase.fasta>
```

or for ordered replicon

```
nextflow run gem-pasteur/macsyfinder -profile cluster_apptainer --models "TFF-SF all" --
↳db-type ordered_replicon --sequence-db '<path/to/replicons/*.fasta>' --outdir <my_
↳results>
nextflow run gem-pasteur/macsyfinder -profile cluster_apptainer --models "TFF-SF all" --
↳db-type ordered_replicon --sequence-db 'file1.fasta,file2.fasta,file3.fst' --outdir
↳<my_results>
```

or if you download the macsyfinder repository, or the the workflow with it's configuration file:

```
nextflow run parallel_macsyfinder.nf -profile standard --models "TFF-SF all" --db-type_
↳ordered_replicon --sequence-db 'data/base/split/GCF_*.fasta' --outdir GCF
```

Note

- For *gembase* data the workflow expected one file with several replicons.
- For *ordered_replicon* or *unordered* the workflow expected several files with one replicon per file.

Warning

See the double quotes surrounding the models value `--models "TFF-SF all"` with out quoting macsyfinder will not received the right argument.

Warning

See the (double) quotes surrounding the models value `--sequence-db '<path/to/replicons/*.fasta>'` with out quoting parallel_macsyfinder will not received all files.

Warning

When you analyzed ordered or unordered replicons (`--db-type` set to *ordered_replicon* or *unordered*) the `--out-dir` option is **REQUIRED**.

standard profile

This profile is used if you want to parallelize MacSyFinder on your machine. You can specify the number of tasks in parallel by setting the *queueSize* value You can also fix the number of cpu used by each task (macsyfinder `--worker` option see *macsyfinder options*) by setting the *params.worker* parameter in *nextflow.config*

```
standard {
  executor {
```

(continues on next page)

(continued from previous page)

```

    name = 'local'
    queueSize = 4
  }
  process {
    errorStrategy = 'ignore'
    withName: macsyfinder {
      cpus = params.worker
    }
  }
}

```

Almost options available in non parallel version are also available for the parallel one. except: * `--db-type` which is set to *gembase* (only data type supported for the parallelized macsyfinder version). * `--out-dir` which is not available.

A typical command line will be:

```

./parallel_macsyfinder.nf -profile standard --models "TFF-SF all" --sequence-db <path/to/
my/gembase.fasta>

```

Note

The options starting with one dash are for nextflow workflow engine, whereas the options starting by two dashes are for macsyfinder workflow.

If you execute this line, 2 kinds of directories will be created.

- One named *work* containing lot of subdirectories this for all jobs launch by nextflow.
- Directories named *merged_macsyfinder_results_XXX* where XXX is the name of the gembase file. This directory contain the final results as in non parallel version.

standard_apptainer or standard_docker profile

If you have not installed *macsyfinder* but you use it through a container docker or <https://apptainer.org/> (former *singularity*) We provide profiles for these situations. With the command line below nextflow will download parallel_macsyfinder from github and download the macsyfinder image from the docker-hub (<https://hub.docker.com/r/gempasteur/macsfinder>) (and apptainer convert the image on the right format on the fly) so you haven't to install anything except nextflow and apptainer or docker.

```

standard_apptainer {
  executor {
    name = 'local'
    queueSize = 4
  }
  process {
    errorStrategy = 'ignore'
    container = 'docker://gempasteur/macsfinder:latest'
    withName: macsyfinder {
      cpus = params.worker
    }
  }
}
singularity {

```

(continues on next page)

(continued from previous page)

```

        enabled = true
    }
}

standard_docker {
    executor {
        name = 'local'
        queueSize = 4
    }
    process {
        errorStrategy = 'ignore'
        container = 'macsyfinder'
        withName: macsyfinder {
            cpus = params.worker
        }
    }
    docker {
        enabled = true
        runOptions = '--user $(id -u):$(id -g)'
    }
}

```

The execution is similar than for installed macsyfinder

```

./parallel_macsyfinder.nf -profile standard_apptainer --models "TFF-SF all" --sequence-
↳db <path/to/my/gembase.fasta>

```

or

```

./parallel_macsyfinder.nf -profile standard_docker --models "TFF-SF all" --sequence-db
↳<path/to/my/gembase.fasta>

```

cluster profile

The cluster profile is intended to work on a cluster managed by SLURM. If your cluster is managed by an other drmm replace executor name by the right value (see [nextflow supported cluster](#))

You can also manage

- The number of tasks in parallel with the *executor.queueSize* parameter (here 500). If you remove this line, the system will send in parallel as many jobs as there are replicons in your data set.
- The queue (or partition in *Slurm* terminology) with *process.queue* parameter (here *common,dedicated*)
- and some options specific to your cluster management systems with *process.clusterOptions* parameter

```

cluster {
    executor {
        name = 'slurm'
        queueSize = 500
    }

    process {
        errorStrategy = 'ignore'
    }
}

```

(continues on next page)

(continued from previous page)

```

    queue = 'common,dedicated'
    clusterOptions = '--qos=fast'
    withName: macsyfinder {
        cpus = params.worker
    }
}

```

To run the parallel version on cluster, for instance on a cluster managed by slurm, I can launch the main nextflow process in one slot. The parallelization and the submission on the other slots is made by nextflow itself. Below a command line to run `parallel_macsyfinder` and use 3 cpus per macsyfinder task, each macsyfinder task can be executed on different machine, each macsyfinder task claim 2 cpus/cores (cpu in *nextflow* terminology/ cores for hardware) to speed up the genes search.

```

sbatch --qos fast -p common nextflow run parallel_macsyfinder.nf -profile cluster --
--models "TFF-SF all" --sequence-db <path/to/my/gembase.fasta> --worker 3

```

The results will be the same as describe in local execution.

cluster_apptainer profiles

You can also use the macsyfinder apptainer image on a cluster, for this use the profile *cluster_apptainer*.

```

sbatch --qos fast -p common nextflow run gem-pasteur/macsfinder -profile cluster_
--apptainer --models "TFF-SF all" --sequence-db <path/to/my/gembase.fasta>

```

In the case of your cluster cannot reach the world wide web. you have to download the singularity image

```

apptainer pull --name macsyfinder.simg docker://gempasteur/macsfinder

```

Then move the image on your cluster modify the nextflow.config to point on the location of the image, and adapt the cluster options (executor, queue, ...) to your architecture

```

cluster_apptainer {
    executor {
        name = 'slurm'
        queueSize = 500
    }

    process {
        errorStrategy = 'ignore'
        container = '/path/to/macsfinder.simg'
        queue = 'common,dedicated'
        clusterOptions = '--qos=fast'
        withName: macsyfinder {
            cpus = params.worker
        }
    }
    singularity {
        enabled = true
        runOptions = '-H $HOME -B /pasteur'
        autoMounts = false
    }
}

```

(continues on next page)

(continued from previous page)

```
}
}
```

then run it

```
sbatch --qos fast -p common nextflow run ./parallel_macsyfinder.nf -profile cluster_
↳ apptainer --models "TFF-SF all" --sequence-db <path/to/my/gembase.fasta>
```

If you want to have more details about the jobs execution you can add some options to generate report:

Execution report

To enable the creation of this report add the `-with-report` command line option when launching the pipeline execution. For example:

```
nextflow run ./parallel_macsyfinder.nf -profile standard -with-report [file name] --
↳ models "TFF-SF all" --sequence-db <path/to/my/gembase.fasta>
```

It creates an HTML execution report: a single document which includes many useful metrics about a workflow execution. For further details see <https://www.nextflow.io/docs/latest/tracing.html#execution-report>

Trace report

In order to create the execution trace file add the `-with-trace` command line option when launching the pipeline execution. For example:

```
nextflow run ./parallel_macsyfinder.nf -profile standard -with-trace --models "TFF-SF
↳ all" --sequence-db <path/to/my/gembase.fasta>
```

It creates an HTML timeline for all processes executed in your pipeline. For further details see <https://www.nextflow.io/docs/latest/tracing.html#timeline-report>

Timeline report

To enable the creation of the timeline report add the `-with-timeline` command line option when launching the pipeline execution. For example:

```
nextflow run ./parallel_macsyfinder.nf -profile standard -with-timeline [file name] --
↳ models "TFF-SF all" --sequence-db <path/to/my/gembase.fasta> ...
```

It creates an execution tracing file that contains some useful information about each process executed in your pipeline script, including: submission time, start time, completion time, cpu and memory used. For further details see <https://www.nextflow.io/docs/latest/tracing.html#trace-report>

Warning

When you run parallelize version of macsyfinder the hhm score for each genes can be different than in non parallel version. As hmmsearch use the size of the sequence database to compute the score.

1.1.2 MacSyFinder functioning

Macromolecular models

MacSyFinder relies on the definition of models of macromolecular systems as a **set of models' components** to be searched by similarity search, and a **set of rules** regarding their genomic organization and their requirement level to make a complete system (mandatory, accessory components, number of components required).

See [below](#) for more details on MacSyFinder's modelling scheme and the section on [Functioning](#) for the principles of the MacSyFinder's search engine.

A **MacSyFinder model** (macy-model for short) is the association of several elements:

- a **definition** which describes the system to detect with a specific **XML grammar** that is described [below](#).
- a set of *HMM profiles* (one per component/gene in the model) to enable the similarity search of the systems' components with the HMMER program.

The models are grouped by *family* possibly gathering *sub-families* (multiple levels allowed), for instance *Secretion*, *Cas-proteins*... A set of models from a same family (coherent set) of systems to detect is called hereafter a **macy-model package** NEW in V2.

Note

For details on how to create your own macy-models, have a look at the [Modeller Guide](#).

Installing models

How to install new models

MacSyFinder does not provide models. You must install models before using it. The `msf_data` utility tool is shipped with *MacSyFinder* to deal with macy-models:

```
msf_data <subcommand> [options]
```

The main sub-commands are

- `msf_data available` to get the list of macy-models available
- `msf_data search` to search a model given its name or a pattern in its description
- `msf_data install` to install a macy-model package (the installed version can be set see `--help`)
- `msf_data cite` to retrieve information on how to cite the model
- `msf_data definition` to display one or a set of model definition
- `msf_data --help` to get the extended list of available subcommands
- `msf_data <subcommand> --help` to get help about the specified subcommand

`msf_data` is NEW in V2

Where the models are located

MacSyFinder looks at several locations to find macy-models.

system-wide installation

By default *msf_data* installs models in a shared location (set by `--install-data` option) that is `/usr/share/macsyfinder/` or `/usr/local/share/macsyfinder` depending on your Operating System distribution. If you use a *virtualenv*, the shared resources are located in the `<virtualenv>/share/macsyfinder` directory.

user-wide installation

If you don't own rights to install system-wide, you can install models in the MacSyFinder's cache located in your home: `$HOME/.macsyfinder/data/`. *msf_data* installs packages in this location when you use the `--user` option. The packages installed in user land is added to the system-wide packages.

Note

If two packages have the same name, the package in the user land supersedes the system-wide package.

project-wide installation

If you cannot install macsy-model packages in system or user land locations, you can install models in specific directory with the `--target` option.

```
msf_data install --target <my_models>
```

The specify this specific location with the `--models-dir` *command-line option*.

```
macsyfinder --db-type ordered_replicon --models-dir=my_models --models TFF-SF all --sequence-db my_genome.fasta
```

The path must point at a directory that contains macsy-model packages as described *above*.

MacSyFinder's search engine

Functioning overview

MacSyFinder is run from the command-line using a variety of input files and options. See *Input dataset* for more details. Below follows a description of its overall functioning.

A. Searching for Systems' components

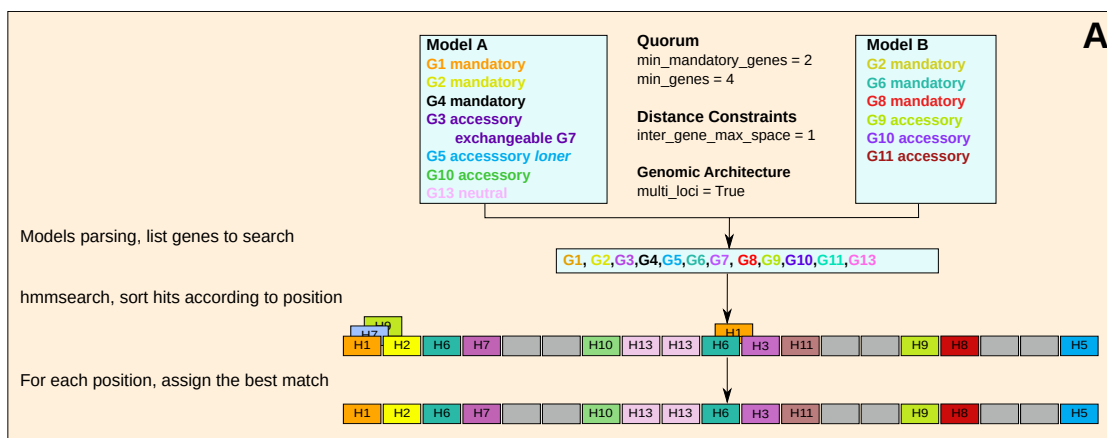
Initially, MacSyFinder **searches for the components** of the *System(s)* to detect by sequence similarity search.

1. From the list of *System(s)* to detect, a **non-redundant list of components to search** is built. For each system, the list can include:

- mandatory components
- accessory components
- neutral components
- forbidden components
- exchangeable components that can be functionally replaced by other components (usually by analogs or homologs). These other components are thus also added to the list of components to search.

See *here for more details on writing MacSyFinder's models*.

2. HMMER is run on the corresponding set of components' HMM profiles, and the hits are filtered according to the criteria defined by the user or by default. This step, and the extraction of significant hits can be performed in parallel (`-w` command-line option). See the *Command-line options*, and the *MacSyLib documentation* for more details.



B. Hits browsing

The following steps depend on whether the input dataset is **ordered** (complete or nearly complete genome(s)), or **unordered** (metagenomes, or unassembled genome(s)) (see the [Input dataset](#) section).

In the case of **ordered datasets** (*ordered_replicon* or *gembase* search mode), the hits are filtered to keep only hits related to the system's model we are looking for. These hits are used to build **clusters of co-localized genes** as defined in the *macsy-model files*. These clusters are then screened to check for the model specifications such as the minimal quorum of “Mandatory” or “Accessory” genes, or the absence of “Forbidden” components.

When the **gene order is unknown** (*unordered* search mode) the power of the analysis is more **limited**. In this case, the presence of systems can only be suggested on the basis of the **quorum** of components - and not based on genomic context information.

For ordered datasets: building clusters of components

The following two steps are reiterated for each model being searched.

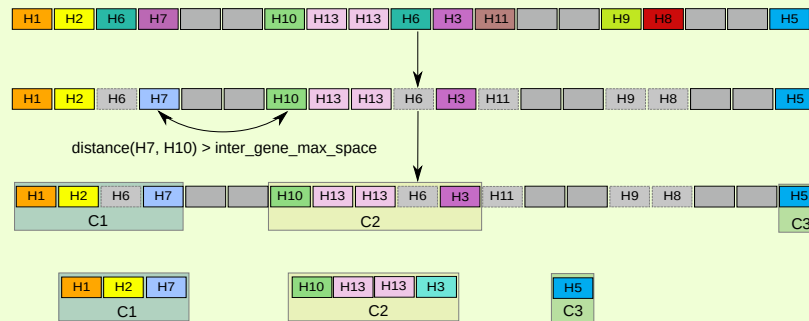
1. The search starts with the filtering of hits to only keep the **hits that are listed in the model** (mandatory, accessory, neutral, forbidden, exchangeable).
2. MacSyFinder searches for sets of contiguous hits to build **clusters**, following the (**co-localization criterion**) for each replicon, as defined in the MacSyFinder's model. Two hits are deemed contiguous if their genomic location is separated by less than d protein-encoding genes, d being the maximum of the two *inter_gene_max_space* parameters from the two genes with hits (system-wise, or gene-specific parameter). The *loner* components may form a cluster on their own.

B - Scanning components

Step 1

Consider the first Model (A) to filter hits
(Genes of model (A): G1, G2, G3, G4, G5, G7, G10, G13)

Build clusters "C" with
co-localizing sets of Hits



Step 2

Check quorum:

- from clusters only
("single_locus" search mode)

- from combinations of clusters
("multi_loci" search mode)

C1	=> Rejected (min_genes_required)
C2	=> Rejected (min_mandatory_genes_required / min_genes_required)
C3	=> Rejected (min_mandatory_genes_required / min_genes_required)
C1 C2	=> System (System A #1: "SA_1")
C1 C3	=> System ("SA_2")
C2 C3	=> Rejected (min_mandatory_genes_required)
C1 C2 C3	=> System ("SA_3")

C1; C2; C3 } rejected_candidates.txt/tsv
C2 C3 }

SA_1: C1 C2
SA_2: C1 C3
SA_3: C1 C2 C3 } all_systems.txt/tsv

Once performed for each model searched, the *next step* is performed.

Note

The clusters that do not fulfill the quorum requirements are stored in the *rejected_candidates.txt/tsv* file.

Note

If several hits which co-locate have the same gene in the model. MSf does not consider them as a cluster.

Note

If a group of gene which co-locate is composed solely of Neutral genes, It has not considered by MSf as a cluster.

For *unordered* datasets:

For each model being searched:

1. The Hits are filtered by model.
2. They are used to check if they reach the quorum (i.e., the clustering step is skipped as there is no notion of genetic distance in this search mode).
3. For each system, if the quorum is reached, hits are reported in the *all_systems.tsv* output file. It has to be noted that forbidden components are listed too, as they can also be informative for the user.

Note

The "unordered" mode of detection is less powerful, as a single occurrence of a given model is filled for an entire dataset with hits that origin is unknown. Please consider the assessment of systems with caution in this mode.

For unordered datasets, the **search so ends**, and MacSyFinder generates the final *output files*.

C. Computing candidate Systems' scores (ordered mode)

This step only applies to the most powerful search mode, i.e., on **ordered datasets**. The whole step is **NEW** in V2

The **new search engine** implemented since version 2.0 of MacSyFinder better explores the space of possible Solutions regarding the presence of Systems in replicons analysed. It creates clusters of hits for Systems' components separately for each System searched, and therefore might find **candidate occurrences of Systems that overlap** in terms of components. Moreover, if a System is possibly encoded at several locations on the replicon analysed (option *multi_loci* set to "True" in the model), this calls for a **combinatorial screening** of the different clusters to assemble them into coherent systems regarding the macsy-models.

- For a given model, clusters are used to "fill up" Systems' occurrence(s) according to the **quorum criteria** defined in the System's model (see function `macsypy.system.match()`):

The *min_genes_required* and *min_mandatory_genes_required* thresholds must be reached.

- In the case of the *single-locus system* search mode (default), each cluster in addition to potential loners are evaluated for System's assessment separately.
- In the case of the *multi-loci system* search mode (`multi_loci=True`), each possible combination of clusters is confronted to the quorum of the System being examined.

The sets of clusters that fulfill the quorum are reported as candidate Systems in the *all_systems.txt* and *all_systems.tsv* output files (see *Output format*), and they obtain a **System's score** (see below).

The clusters that do not allow to form a candidate System are reported in the *rejected_candidates.txt* and *rejected_candidates.tsv* output files.

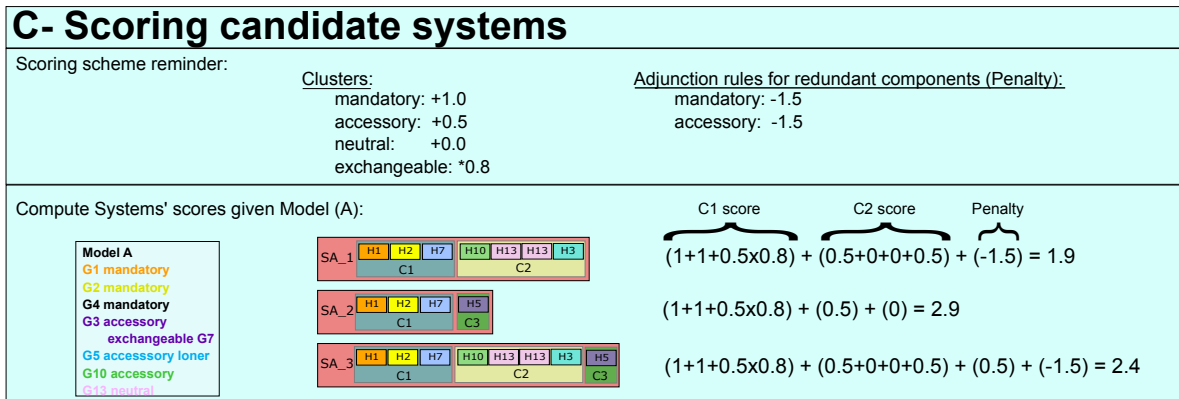
- We introduce a **scoring scheme for candidate Systems**, to easily separate combinations of clusters that are readily more similar to a system's model than others.

The assumptions behind this scoring scheme are the following:

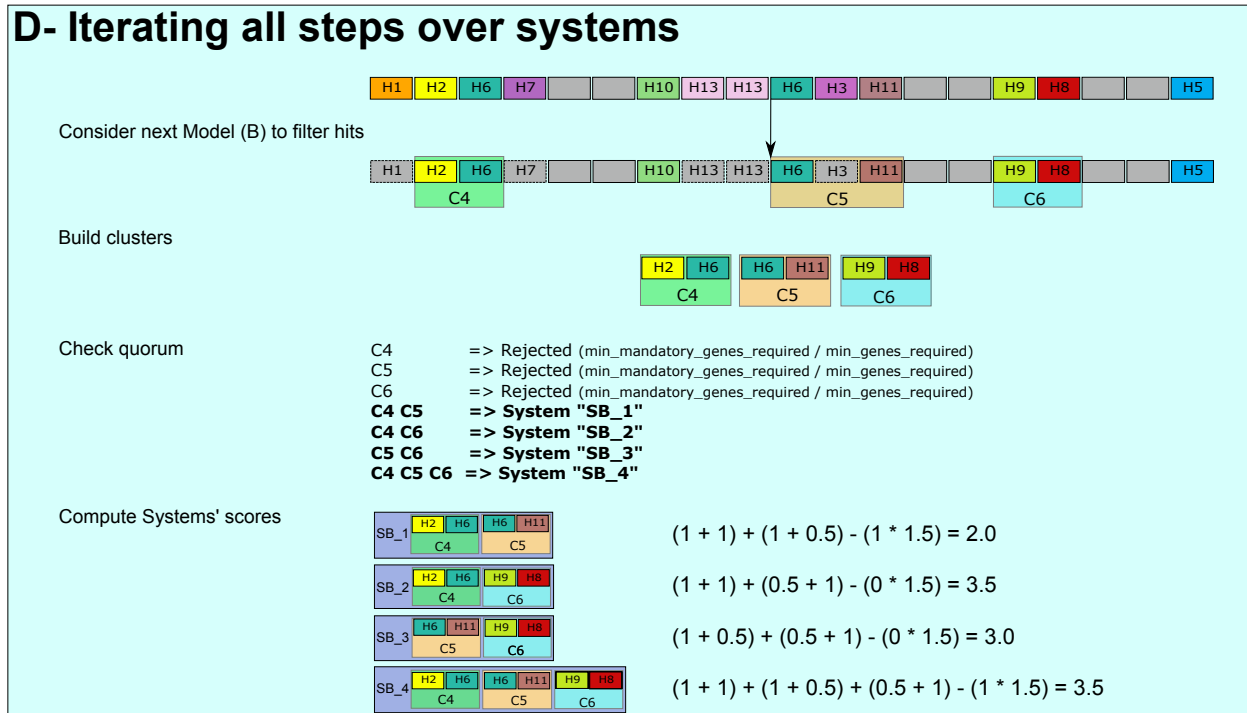
- We set a score for the different types of genes/components when defining a **cluster's score**. Here are the default values, but these *can be changed*:
 - * +1.0 is added when a *mandatory* gene is present
 - * +0.5 is added when an *accessory* gene is present
 - * +0.0 is added when a *neutral* gene is present
 - * *0.8 (a factor of 0.8) is applied to the above-scores when the function is fulfilled by an *exchangeable* gene
 - * *0.7 (a factor of 0.7) is applied to the above-scores if the gene is a *loner* and *multi system* component.
- When combinations of clusters are explored in order to fulfill macsy-models' requirements and build candidate systems ("multi_loci" mode, several clusters can make a complete *System*), we sum the score of clusters to assign a *System's* score.
- In addition, we want to **favor concise sets of clusters** to fulfill a *System's* model. We thus **penalize the adjunction of a cluster** to a candidate *System* when this cluster does not bring any new components to the *System's* quorum, or when it brings **redundant components**. Thus:
 - * -1.5 is added when a **redundant** mandatory gene is added when adjuncting the cluster to a candidate *System*
 - * -1.5 is added when a **redundant** accessory gene is added when adjuncting the cluster to a candidate *System*

- * for the components that are *loner* and *multi system*, the score of the loner component is added only if the function is not fulfilled in the other clusters. In this case, even if there are several occurrences of the component, it is counted only once (and no penalty is applied).
- Only candidate sets of clusters that fulfill a macsy-model and that are thus designated candidate *Systems*, obtain a **System's score**

In summary, a Systems's score is made of two parts: the **sum of the scores** of the Clusters it is made of, plus a **penalty part** to avoid too much component's redundancy in Cluster's combinations. The systems' scoring step is exemplified in this figure:



D. Repeat operations B and C for the other models being searched



This search for candidate *Systems* from different models results in a number of possible *Solutions* representing combinations of putative sets of *Systems* in the analysed dataset.

E. Computing possible Solutions, defining the best one (ordered mode)

At the end of the previous step MacSyFinder has computed all potential *Systems* present in the replicon, made of combinations of Clusters and *loner* components that fulfill the model's requirements, which are themselves made of a subset of Hits (remember, Hits are at 1st filtered and treated separately for each model of System to be detected). Candidate *Systems* may thus overlap by being partly made of the same components, or even partly being made of the same Clusters.

We define a *Solution* as being a **set of compatible Systems**, i.e. that do not have any overlaps between their components. All possible *Solutions* are combinatorially explored and consist in all possible sets of compatible *Systems*.

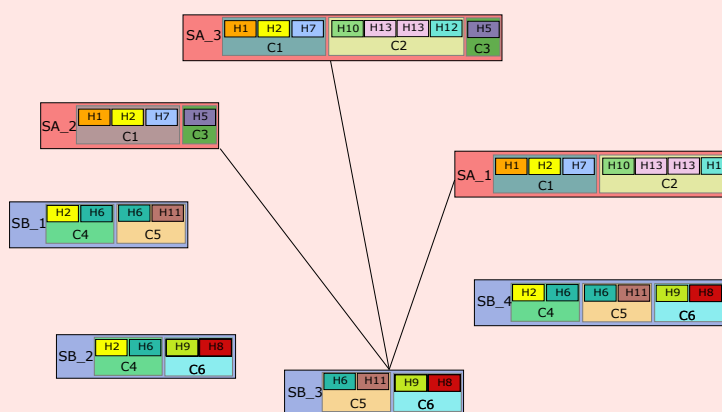
A scoring scheme enables to separate between sets of *Solutions*. A **Solution's score** is basically the **sum of its Systems' scores**. The overall procedure of exploring the space of all possible *Solutions* while finding the optimal one, i.e. that with the maximal score, is performed at once using a graph solution to this problem, implemented in the `networkx` package.

We create a graph where each potential *System* is a vertex, and we create an edge between pairs of vertices if they do not share any components (compatible *Systems*). Once the graph is created we look for the **maximum clique** which maximizes the score. This allows to provide the user with one, or multiple *Solutions* that have the **best score possible** among all combinations of compatible *Systems*.

E- Computing solutions

Step 1

Build a graph of Systems
(edges between compatible Systems)



Step 2

Compute the scores of maximal cliques

SA_3	<div><div>H1H2H7C1</div><div>H10H13H13H12C2</div><div>H5C3</div></div>	SB_3	<div><div>H6H11H9H8C5</div><div>C6</div></div>	4.25 + 3.0 = 7.25	} best_solution.tsv all_best_solutions.tsv
SA_2	<div><div>H1H2H7H5C1</div><div>C3</div></div>	SB_3	<div><div>H6H11H9H8C5</div><div>C6</div></div>	3.25 + 3.0 = 6.25	
SB_1	<div><div>H2H6H6H11C4</div><div>C5</div></div>			2.0	
SB_2	<div><div>H2H6H9H8C4</div><div>C6</div></div>			3.5	
SB_3	<div><div>H6H11H9H8C5</div><div>C6</div></div>	SA_1	<div><div>H1H2H7H10H13H13H12C1</div><div>C2</div></div>	3.0 + 3.75 = 6.75	
SB_4	<div><div>H2H6H6H11H9H8C4</div><div>C5</div><div>C6</div></div>			3.5	

1.1.3 Frequently Asked Questions

Frequently Asked Questions

How to report an issue?

If you encounter a problem while running MacSyFinder, please submit an issue on the dedicated page of the [GitHub project](#)

To ensure we have all elements to help, please provide:

- a concise description of the issue
- the expected behavior VS observed one
- the exact command-line used
- the version of MacSyFinder used
- the exact error message, and if applicable, the *macsyfinder.log* and *macsyfinder.conf* files
- if applicable, an archive (or link to it) with the output files obtained
- if possible, the smallest dataset there is to reproduce the issue
- if applicable, this would also include the macy-models (XML models plus HMM profiles) used (or precise version of the models if there are publicly available). Same as above, if possible, please provide the smallest set possible of models and HMM profiles.

All these will definitely help us to help you! ;-)

How to cite MacSyFinder and published macy-models?

- [Abby et al. 2014](#), *PLoS ONE* for the **general principles of MacSyFinder** (version 1), and the corresponding set of Cas systems (CasFinder, 1st version).
- [Abby and Rocha 2012](#), *PLoS Genetics*, for the study of the evolutionary relationship between the T3SS and the bacterial flagellum, and how were designed the corresponding HMM protein profiles.
- [Abby et al. 2016](#), *Scientific Reports*, for the description of bacterial protein secretion systems' models (TXSScan: T1SS, T2SS, T5SS, T6SS, T9SS, Tad, T4P).
- [Denise et al. 2019](#), *PLoS Biology*, for the description of type IV-filament super-family models (TFF-SF: T2SS, T4aP, T4bP, Com, Tad, archaeal T4P).
- [Rendueles et al. 2017](#), *PLoS Pathogens*, for the CapsuleFinder set of models.
- [Couvin, Bernheim et al. 2018](#), *Nucleic Acids Research*, for the updated version of the set of Cas systems' models, CasFinder.

What do MacSyFinder command lines look like?

Here are a few examples of command line formation:

To browse interactive help:

```
macsyfinder -h
```

The minimal command line, to search all systems with models from the “TFF-SF” set of models (installed with *msf_data*):

```
macsyfinder --db-type ordered_replicon --sequence-db genome.fasta --models
TFF-SF all
```

To search for several systems (ModelA and ModelB) from the “model_family” set of models that can be found in the “./my-models” folder:

```
macsyfinder --db-type ordered_replicon --sequence-db genome.fasta --models
model_family ModelA ModelB --models-dir ./my-models
```

To alter the search parameters and allow a maximal distance between components of 20 for the T2SS and 15 for the Tad pilus:

```
macsyfinder --db-type ordered_replicon --sequence-db genome.fasta --models  
TFF-SF all --inter-gene-max-space T2SS 20 --inter-gene-max-space Tad 15
```

To alter the search parameters and allow the Tad pilus to be made of multiple loci:

```
macsyfinder --db-type ordered_replicon --sequence-db genome.fasta --models  
TFF-SF all --multi-loci Tad
```

In *gembase* or *ordered_replicon* mode *macsyfinder* need to index the sequence-db. By default, this index is write beside the sequence-db file. But sometimes the directory where the sequence-db is located is not writable, in centralized shared data in multi user environnement for instance. To avoid to copy sequences in other location, you could specify an alternate directory for the index with `--index-dir` (This directory must exists):

```
macsyfinder --db-type ordered_replicon --sequence-db genome.fasta --index-dir  
my-indexes --models TFF-SF all
```

See also the [MacSyFinder Quick Start](#) section for more examples.

What search mode to be used?

Depending on the type of dataset you have, you will have to adapt MacSyFinder's search mode.

- If you have a fasta file from a complete genome where **proteins are ordered** according to the corresponding genes' order along the replicon, your dataset is entitled to the most powerful search mode (see below): *ordered_replicon* and use the following option `--db-type ordered_replicon`.
- If you have a fasta file of proteins with **no sense of the order** of the corresponding genes along the chromosome(s) or replicon(s), you will have to use the *unordered* search mode with the following option: `--db-type unordered`
- If you have **multiple ordered replicons** to analyse at once, you can follow the *Gembase* convention to name the proteins in the fasta file, so that the original replicons can be assessed from their name: [see here for a description](#).

Note

- When the **gene order is known** (*ordered_replicon* search mode) the power of the analysis is **maximal**, since both the genomic content and context are taken into account for the search.
- When the **gene order is unknown** (*unordered* search mode) the power of the analysis is more **limited** since the presence of systems can only be suggested on the basis of the quorum of components - and not based on genomic context information.

More on command-line options [here](#) and on MacSyFinder's functioning [here](#).

How to deal with fragmented genomes (MAGs, SAGs, draft genomes)?

There are more and more genomes available which are not completely assembled, or are fragmented and incomplete. In this case, several options can be considered.

1. If your genome is at least partially assembled and contigs are not too short, you might “feel lucky” and first consider to run MacSyFinder with the *ordered_replicon* mode. It could be particularly efficient if you are investigating systems encoded by compact loci (Cas systems, some secretion systems...), as they might be encoded by a single contig.
2. On top of the *ordered_replicon* mode, you might add the option “multi-loci” to the systems to annotate (if not already the case), in order to maximize the chance to annotate an entire system, even if encoded across several contigs.

3. The *unordered* mode can be used in complement of the two above options, e.g. to retrieve some of the missing components. It will enable to assess the genetic potential and possible presence of a system, independently of the quality of assembly of the genome. It might also be the only reasonable option if the genome is too fragmented and/or too incomplete.

Note

- The results obtained with the *ordered_replicon* mode on a fragmented genome have to be considered carefully, especially with respect to the contigs' borders, as some proteins from different contigs might be artificially considered as closely encoded.
- To retrieve “fragments” of a system not found to reach the quorum in the *ordered_replicon* mode, it is possible to retrieve clusters of genes from the *rejected_candidates.tsv* file.

How to interpret the results from an *unordered* search?

As mentioned above, in the *unordered* search mode, the inference of a system's presence is only based on the list of components found in the protein dataset. Thus, the kind of search specificity provided when using the genomic context (components next to each other are more likely to be part of a same system's occurrence) is not within reach.

In the *unordered* search mode, the number of proteins selected as system's components (based on the filtering of HMM profiles' similarity search) is reported. We decided to report all kinds of system's components, including the *forbidden* ones in order to raise awareness of the user -> even if all constraints are met for the system's inference (here, the quorum: minimal number of components), it cannot be excluded that a *forbidden* component would lie next to the *bona fide* components (*mandatory* and *accessory* ones) in the genome...

In the end, the *unordered* search mode provides an idea as to whether the **genetic potential** for a given system is found in the set of proteins analysed, with no attempt to assign proteins to particular systems' occurrences, nor guarantee as to whether *forbidden* components should be considered for the potential occurrences.

How to search for multiple systems at once?

- It is possible to search for only some systems from a macsy-model package. In this case, the command-line should be formed as follows:

```
macsyfinder --models TXSS Flagellum T2SS --sequence-db mygenomes.fasta --db-type gembase
```

This would run the search of the systems “Flagellum” and “T2SS” in the dataset “mygenomes.fasta”.

- To run the search of all the models contained in a macsy-model package, use the following:

```
macsyfinder --models TXSS all --sequence-db mygenomes.fasta --db-type gembase
macsyfinder --models CRISPRCas all --sequence-db mygenomes.fasta --db-type gembase
macsyfinder --models CRISPRCas/typing all --sequence-db mygenomes.fasta --db-type gembase
```

You can see that the *all* keyword can not only be applied to an entire macsy-model package and its entire hierarchy, but can also be ran on all the systems from a macsy-model sub-directory.

When can the option *-previous-run* be used?

The option *-previous-run* enables to avoid running the HMM profile search and the hits extraction when the set of systems to search and the replicons to analyse are exactly the same between runs. This enables to alter the features of the systems to be searched for, i.e. basically any feature found in the XML file of the corresponding models:

- the maximal distance allowed between components to be considered as part of a same locus *-inter-gene-max-space*

- the minimal number of components to be found to infer a full system *–min-mandatory-genes-required* and *–min-genes-required*
- the general genomic architecture of the system *–multi-loci*

This also means that there are a number of options that are incompatible with *–previous-run*, including:

```
--config, --sequence-db, --profile-suffix, --res-extract-suffix, --e-value-res, --db-  
↪ type, --hmmer
```

Which output file to be used to get ONE solution?

Since version 2 of MacSyFinder, a combinatorial exploration of the possible sets of systems is performed. A scoring scheme has been set up to differentiate between solutions, in order to provide the user with the most complete set of systems as possible given the searched models. This score is maximal for the “best solution”. This also means that some solutions might get the same maximal score. In this case, one can wonder how to find all the equivalent solutions, and another, how to simply pick one solution among the best, whichever it is. We thus propose several kind of *output files*.

- All equivalent best solutions are found in the *all_best_solutions.tsv* file.
- One best solution is given in the *best_solution.tsv* file.

Note

For those more familiar with the output files from MacSyFinder v1, the file *best_solution.tsv* is the closest from the previous output file *macsyfinder.report*.

Where to find MacSyFinder models?

Since version 2, there is a tool to enable the download and installation of published models from a repository: the *msf_data* tool.

See [here for details](#) on how to use it.

What are the rules for options precedence?

MacSyFinder offers many ways to parametrize the systems’ search: through the command-line, through various configuration files (for the models, for the run, etc...). It offers a large control over the search engine. But it also means you can get lost in configuration. ;-)

Here is a recap of the rules for options precedence. In a general manner, the command line always wins.

The precedence rules between the different levels of configuration are:

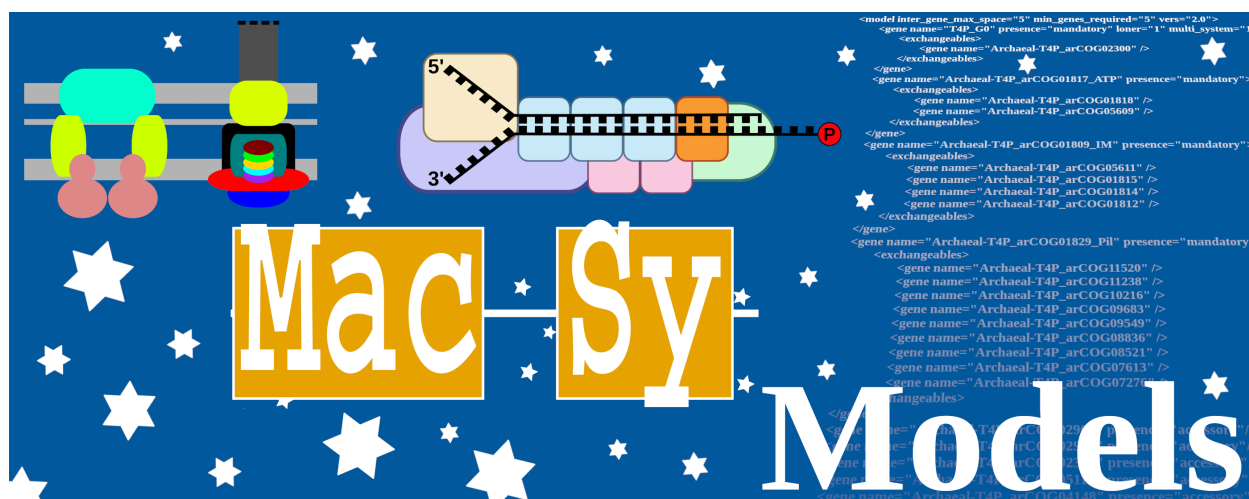
```
system < home < model < project < --cfg-file | --previous-run < command line options
```

- **system:** the *macsyfinder.conf* file either in */etc/macsyfinder/* or in *\${VIRTUAL_ENV}/etc/macsyfinder/* in case of a *virtualenv* this configuration affects only the MacSyFinder version installed in this *virtualenv*
- **home:** the *~/macsyfinder/macsyfinder.conf* file
- **model:** the *model_conf.xml* file at the root of the model package
- **project:** the *macsyfinder.conf* file found in the directory where the *macsyfinder* command was run
- **cfgfile:** any configuration file specified by the user on the command line (conflicts with the *–previous-run* option)

- **previous-run:** the *macsyfinder.conf* file found in the results directory of the previous run (conflicts with the *-cfg-file* option)
- **command line:** any option specified directly in the command line

MODELLER GUIDE

2.1 Modeller Guide



2.1.1 Modelling Systems with MacSyFinder

Installation

MacSyFinder works with models for macromolecular systems that are not shipped with it, you have to install them separately. See the *msf_data* section below. We also provide container so you can use macsyfinder directly.

MacSyFinder Installation procedure

To develop new models and share them, MacSyFinder requires *git* and the python librarie *GitPython*

Below the procedure to install *MacSyFinder* in *modeler* mode in a virtualenv

```
python3 -m venv macsyfinder
cd macsyfinder
source bin/activate
python3 -m pip install macsyfinder[model]
```

GitPython dependency will be automatically retrieved and installed when using *pip* for installation (see below).

Warning

But you have to install *git* by yourself (using your preferred package manager)

From Conda/Mamba

From version 2.0 and above, MacSyFinder is packaged for Conda/Mamba. The Conda/Mamba package includes modeler dependencies.

From container

From version 2.0 and above, a docker image is available. This image allows you to develop models.

Models installation with *msf_data*

Once MacSyFinder is installed, you have access to an utility program to manage the models: *msf_data*

This script allows to search, download, install and get information from MacSyFinder models stored on github (<https://github.com/macsy-models>) or locally installed. The general syntax for *msf_data* is:

```
msf_data <general options> <subcommand> <sub command options> <arguments>
```

To list all models available on *macsy-models*:

```
msf_data available
```

To search for models on *macsy-models*:

```
msf_data search TXSS
```

you can also search in models description:

```
msf_data search -S secretion
```

To install a model package:

```
msf_data install <model name>
```

To install a model when you have not the right to install it system-wide

To install it in your home (*./macsyfinder/data*):

```
msf_data install --user <model name>
```

To install it in any directory:

```
msf_data install --target <model dir> <model_name>
```

To know how to cite a model package:

```
msf_data cite <model name>
```

To show the name of the models and the structure of installed model package:

```
msf_data show <model package name>
```

for instance `msf_data show TXSScan`


```

TXSScan
├--archaea
│   └--Archaeal-T4P
├--bacteria
│   └--diderm
│       ├──Flagellum
│       ├──MSH
│       ├──T1SS
│       ├──T2SS
│       ├──T3SS
│       ├──T4aP
│       ├──T4bP
│       ├──T5aSS
│       ├──T5bSS
│       ├──T5cSS
│       ├──T6SSi
│       ├──T6SSii
│       ├──T6SSiii
│       ├──T9SS
│       ├──Tad
│       ├──pT4SSi
│       └--pT4SSt
└--monoderm
    └--ComM

```

TXSScan (1.1.3) : 19 models

To show the model definition:

```
msf_data definition <package or subpackage> model1 [model2, ...]
```

for instance to show model definitions T6SSii and T6SSiii in TXSS+/bacterial subpackage:

```
msf_data definition TXSS+/bacterial T6SSii T6SSiii
```

To show all models definitions in TXSS+/bacterial subpackage:

```
msf_data definition TXSS+/bacterial
```

To create a git repository with a skeleton for your own model package:

```
msf_data init --pack-name <MY_PACK_NAME> --maintainer <"mantainer name"> --email
↳<maintainer email> --authors <"author1, author2, ..">
```

above `msf_data` with required options. Below I add optional but recommended options.

```
msf_data init --pack-name <MY_PACK_NAME> --maintainer <"mantainer name"> --email
↳<maintainer email> --authors <"author1, author2, .."> \
--license cc-by-nc-sa --holders <"the copyright holders"> --desc <"one line package_
↳description">
```

To list all `msf_data` subcommands:

```
msf_data --help
```

To list all available options for a subcommand:

```
msf_data <subcommand> --help
```

For models not stored in *macsy-models* the commands *available*, *search*, *installation* from remote or *upgrade* from remote are **NOT** available.

For models **NOT** stored in *macsy-models*, you have to manage them semi-manually. Download the archive (do not unarchive it), then use *msf_data* to install the archive.

Models Package

MacSyFinder relies on the definition of models of macromolecular systems as a **set of models' components** to be searched by similarity search, and a **set of rules** regarding their genomic organization and their requirement level to make a complete system (mandatory, accessory components, number of components required).

See the section [The XML hierarchy](#) for more details on MacSyFinder's modelling scheme and the section on [Functioning](#) for the principles of the MacSyFinder's search engine.

A **MacSyFinder model** (*macsy-model* for short) is the association of several elements:

- a **definition** which describes the system to detect with a specific **XML grammar** that is [described here](#).
- a set of *HMM profiles* (one per component/gene in the model) to enable the similarity search of the systems' components with the HMMER program.

The models are grouped by *family* possibly gathering *sub-families* (multiple levels allowed), for instance *Secretion*, *Cas-proteins*... A set of models from a same family (coherent set) of systems to detect is called hereafter a **macsy-model package** NEW in V2.

Structure of a macsy-model package

A macsy-model package follows the following structure:

```
family_name
|_____ metadata.yml
|_____ LICENSE
|_____ README.md
|_____ model_conf.xml
|_____ definitions
|           |_____ model_1.xml
|           |_____ model_2.xml
|           :
|_____ profiles
|           |_____ geneA.hmm
|           |_____ geneB.hmm
|           |_____ geneC.hmm.gz
```

If the package contains sub-families:

```
family_name
|_____ metadata.yml
|_____ LICENSE
```

(continues on next page)

(continued from previous page)

```

|_____ README.md
|_____ model_conf.xml
|_____ definitions
|           |_____ subfamilyA
|           |           |_____ model_1.xml
|           |           |_____ model_2.xml
|           |_____ subfamilyB
|           |           |_____ model_3.xml
|           |           |_____ model_4.xml
|           :
|_____ profiles
|           |_____ geneA.hmm
|           |_____ geneB.hmm
|           |_____ geneC.hmm.gz

```

For examples of macsy-model packages, please visit <https://github.com/macsy-models>

You can create a template for your package by using *msf_data init*. It will create for you:

- the data package directory with the right structure.
- a template of *metadata.yaml*.
- a template of *README.md* file.
- a generic *model_conf.xml* file.
- a LICENSE file if *-license* option is set.
- a COPYRIGHT file if *-holders* option is set.
- a directory *definitions* with an example of model definition (*model_example.xml* to remove before publishing).
- a directory *profiles* where to put the hmm profiles corresponding to the models genes.

Note

MSF can also read *.gz* compressed files; it will uncompress them on the fly. The compressed files must end with the *.gz* extension. For the *hmmsearch* step You need to have *gunzip* installed on your system for this to work.

README.md

A description of the package: what kind of systems the package models, how to use it etc... in *markdown* format. The Readme is displayed to the user on the macsy-models repository on Github. It is also displayed when the user runs *msf_data help*.

LICENSE

The license is used to protect your work when sharing it. If you don't know which license to choose, have a look at [CreativeCommons](#) *This file is optional, but highly recommended.*

Metadata file

The *metadata.yml* file contains some meta information about the package itself.

It is in **YAML** format and must have the following structure:

```
---
maintainer:
  name: The name of the person who maintains/to contact for further information.
  ↪(required)
  email: The email of the maintainer (required)
short_desc: A one line description of the package (can e.g. be used for *msf_data*
  ↪searches) (required)
vers: The package version (DEPRECATED)
cite: The publication(s) to cite by the user when the package is used (optional, used by
  ↪`msf_data cite`)
doc: Where to find extended documentation (optional)
license: The license under the package is released (optional but highly recommended)
copyright: The copyright of the package (optional)
```

For example:

```
---
maintainer:
  name: first name last name
  email: login@my_domain.com
short_desc: Models for 15 types of secretion systems or bacterial appendages (T1SS, T2SS,
  ↪ T3SS, T4P, pT4SSi, pT4SSii, T5aSS, T5bSS, T5cSS, T6SSi, T6SSii, T6SSiii, Flagellum,
  ↪ Tad, T9SS).
cite:
  - |
    Abby Sophie S., Cury Jean, Guglielmini Julien, Néron Bertrand, Touchon Marie, Rocha
    ↪ Eduardo P. C. (2016).
    Identification of protein secretion systems in bacterial genomes.
    In Scientific Reports, 6, pp. 23080.
    http://dx.doi.org/10.1038/srep23080
doc: https://github.com/macsy-models/TXSS
license: CC BY-NC-SA 4.0 (https://creativecommons.org/licenses/by-nc-sa/4.0/)
copyright: 2014-2022, Institut Pasteur, CNRS
```

Note

- - specify an item of yaml list
- | is used to specify a single item but over multiple lines.

Error

This *metadata.yml* file is **mandatory**. Without this file your archive/repository will not be considered as a *macsy-model* package.

Warning

The field *vers* (the package version) is deprecated. *msf_data install* rely only on the git tag.

Model configuration

The modeler has the possibility to specify some options that are specific to their package, different than the MacSyFinder defaults in the *model_conf.xml* file. **NEW in v2**

These options can be grouped in two families: the scoring weights and filtering options.

Scoring weights:

- **mandatory** (*float* default = 1.0)
- **accessory** (*float* default = 0.5)
- **exchangeable** (*float* default = 0.8)
- **loner_multi_systems** (*float* default = 0.7)
- **redundancy_penalty** (*float* default = 1.5)

Filtering options:

- **e_value_search** (*float* default = 0.1)
- **i_evalue_sel** (*float* default = 0.001)
- **profile_coverage** (*float* default = 0.5)
- **cut_ga** (*bool* default = True)

All these options are optional and can be omitted in the configuration file, **the file itself is optional**. The precedence rules between the different levels of configuration are:

```
system < home < model < project < --cfg-file | --previous-run < command line options
```

- **system**: the *macsyfinder.conf* file either in */etc/macsyfinder/* or in *\${VIRTUAL_ENV}/etc/macsyfinder/* in case of a *virtualenv* this configuration affects only the MacSyFinder version installed in this *virtualenv*
- **home**: the *~/macsyfinder/macsyfinder.conf* file
- **model**: the *model_conf.xml* file at the root of the model package
- **project**: the *macsyfinder.conf* file found in the directory where the *macsyfinder* command was run
- **cfgfile**: any configuration file specified by the user on the command line (conflicts with the *-previous-run* option)
- **previous-run**: the *macsyfinder.conf* file found in the results directory of the previous run (conflicts with the *-cfg-file* option)
- **command line**: any option specified directly in the command line

The *model_conf.xml* configuration file is in xml format and must have the following structure:

```
<model_config>
  <weights>
    <mandatory>1</mandatory>
    <accessory>0.5</accessory>
    <exchangeable>0.8</exchangeable>
    <redundancy_penalty>1.5</redundancy_penalty>
```

(continues on next page)

(continued from previous page)

```

    <out_of_cluster>0.7</out_of_cluster>
  </weights>
  <filtering>
    <e_value_search>0.1</e_value_search>
    <i_evalue_sel>0.01</i_evalue_sel>
    <coverage_profile>0.5</coverage_profile>
    <cut_ga>True</cut_ga>
  </filtering>
</model_config>

```

Details about the scoring method can be obtained [here](#).

Macromolecular models

MacSyFinder relies on the definition of models of macromolecular systems as a **set of models' components** to be searched by similarity search, and a **set of rules** regarding their genomic organization and their requirement level to make a complete system (mandatory, accessory components, number of components required).

See [below](#) for more details on MacSyFinder's modelling scheme and the section on [Functioning](#) for the principles of the MacSyFinder's search engine.

A **MacSyFinder model** (macy-model for short) is the association of several elements:

- a **definition** which describes the system to detect with a specific **XML grammar** that is described [below](#).
- a set of *HMM profiles* (one per component/gene in the model) to enable the similarity search of the systems' components with the HMMER program.

The models are grouped by *family* possibly gathering *sub-families* (multiple levels allowed), for instance *Secretion*, *Cas-proteins*... A set of models from a same family (coherent set) of systems to detect is called hereafter a **macy-model package** NEW in V2.

Principles, and how to write macy-models definitions

Macy-models are written as XML files, and should be named with the name of the system to detect as a prefix, and the XML file extension as a suffix. For example, 'T1SS.xml' for T1SS (Type I Secretion System).

A macy-model defines a macromolecular System as:

- A set of **components** (*i.e.* proteins, or protein-coding genes given the context) with different attributes that are used for system's **content description**.
- Features regarding the **genomic architecture** of the systems' components for system detection.
- Rules for **quorum** specifying how many components are required to infer the presence of a complete system.

Macy-model Components

Four distinct **types of components** can be used to model the System's content. Components correspond to Gene objects in MacSyFinder's implementation, and point to corresponding HMM protein profiles.

- **mandatory** components represent components that are essential to be found to infer the system's presence.
- **accessory** components correspond to components that can be found in some systems' occurrence (or quickly evolving components that are hard to detect with a single HMM profile and thus can be missed along similarity search).
- **neutral** components are used to build/extend clusters of proximal genes/components on the replicon analysed, but are not part of the quorum (*i.e.*, not taken into account to assess the system's presence). NEW in V2

- **forbidden** components are components which presence is eliminatory for the system's presence assessment.

Specifying a genomic organization

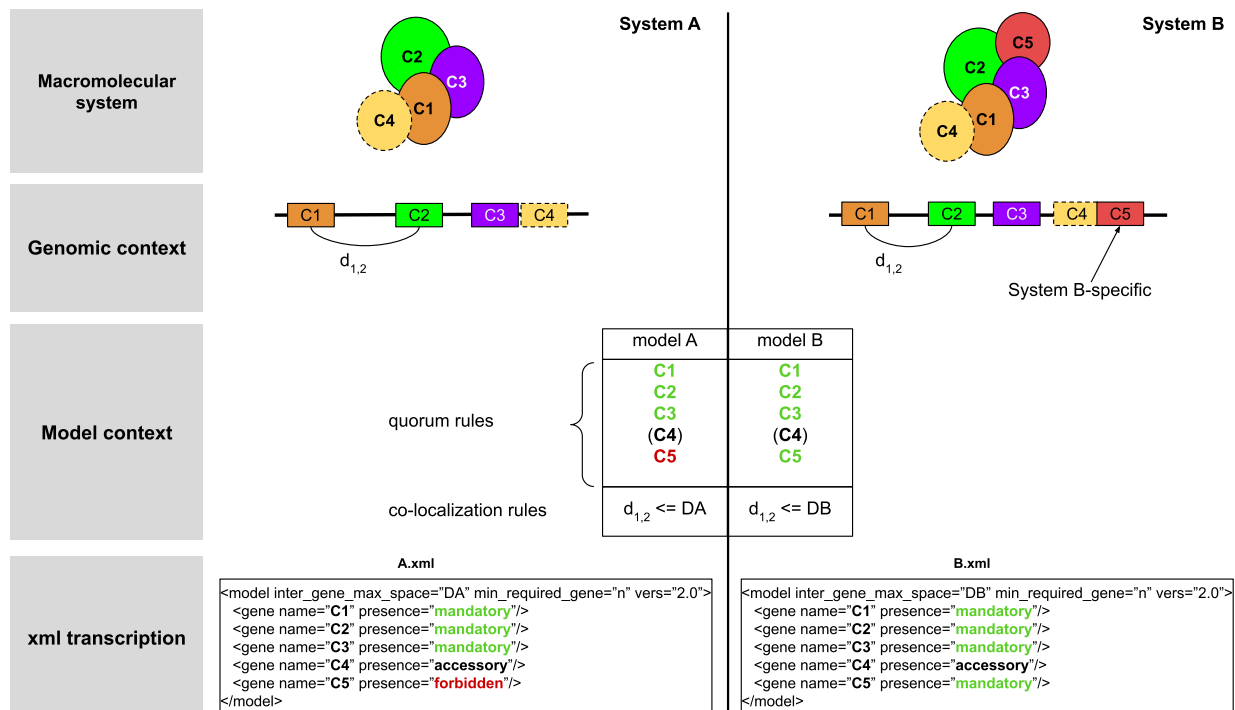
Beyond its list of Components, a MacSyFinder's model of a System is defined by the genomic organization of its components. This genomic organization can be defined in several ways:

- the general System's architecture, whether it is *single-locus* or *multi-loci* (encoded at one or several loci)
- the co-localization criteria defined either at the System level or at the Gene (component) level:
 - the *inter-gene-max-space* parameter (system- or gene- wise)
 - the *loner* parameter (gene- wise)

See [below](#) for more details on how to specify these parameters in a macy-model.

The XML hierarchy

A System's model is defined using a specific XML grammar that is hereby described. It consists in a hierarchic view of a Model that has specific features described through parameters, and is made of a set of Genes that have specific features themselves. All these elements and corresponding parameters will parametrize the search of Systems matching the search by MacSyFinder, in terms of Gene content and genomic architecture criteria.

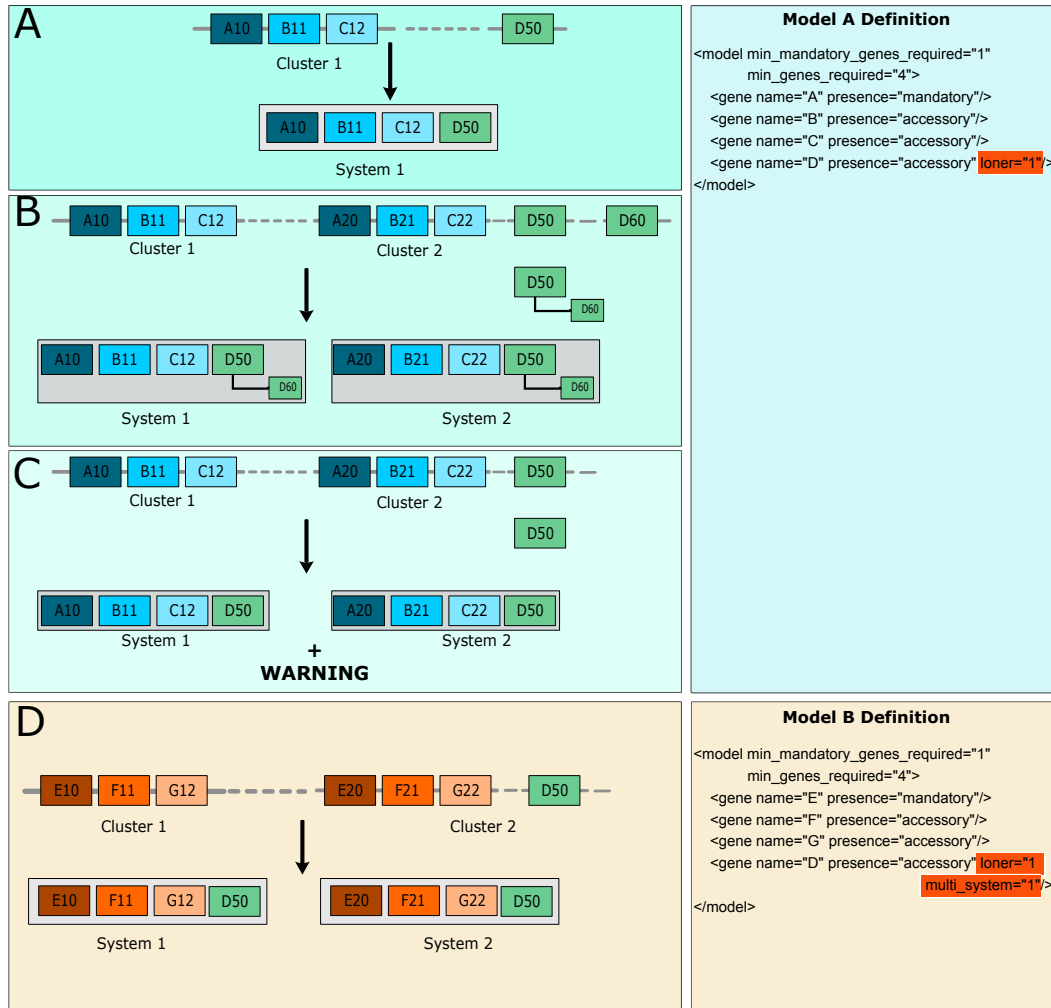


- The element root of a System's model is "model".
 - It has a mandatory attribute: "inter_gene_max_space", an integer representing the maximal number of components without a match between two components with a match for a component profile in order to consider them contiguous (part of a same *Cluster*).
 - The version of the XML grammar (the actual version is "2.0")
 - The element "model" may have attributes:

- * **min_mandatory_genes_required**: an *integer* representing the minimal number of mandatory genes required to infer the system's presence.
 - * **min_genes_required**: an *integer* representing the minimal number of mandatory or accessory genes (whose corresponding proteins match a profile of the model) required to infer the system's presence.
 - * **multi_loci**: a *boolean* set to True ("1", "true" or "True") to allow the definition of "scattered" systems (i.e., systems encoded at different genomic loci or by different gene *clusters*). If not specified, *default value is false*.
 - * **max_nb_genes** define how many genes is necessary to consider a system as full. By default it is the sum of mandatory and accessory genes. But sometimes in special cases, there is 2 profiles, so 2 *msf* genes in model for one real gene. So in system only one gene can be detected and the wholeness is false.
- The model contains one or more element(s) "gene" that correspond(s) to the genetic components of the macromolecular system.
- The element "gene" has several mandatory attributes:
 - **name**: a *string* representing the name of the component/gene which must match that of a profile enclosed in the profile directory of the macsy-model package (see *below*).
 - **presence**: a *string* representing the status of the gene's presence in the system. It can take four values among "mandatory", "accessory", "neutral", "forbidden" (see above).

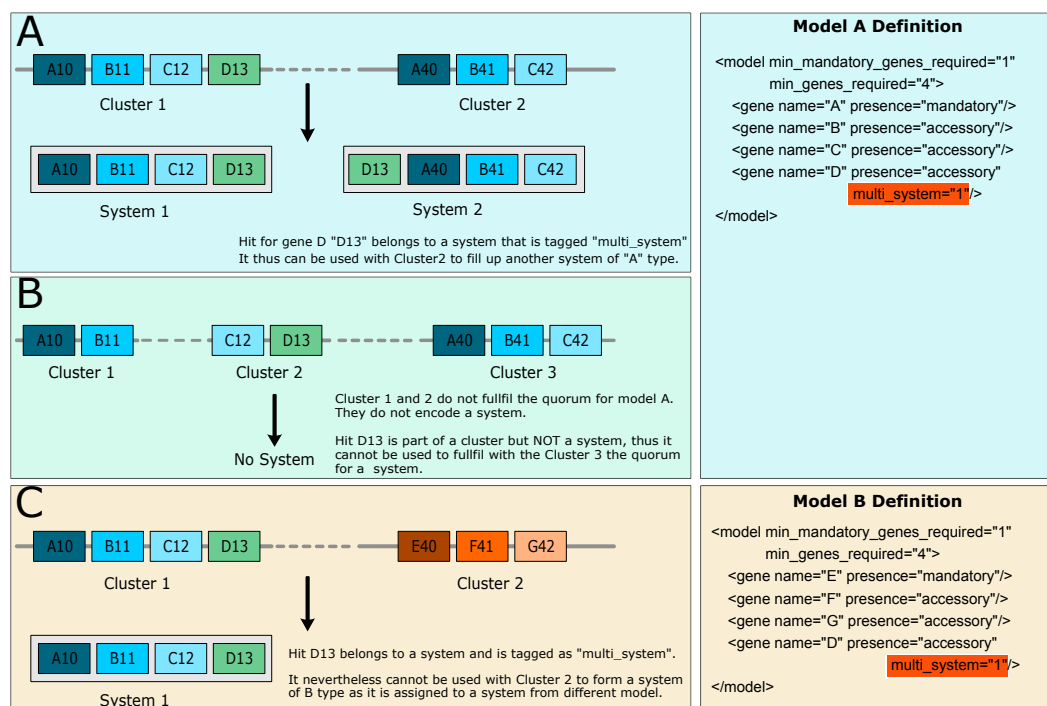
The element "gene" may have other attributes:

- **loner**: a *boolean*. A *loner* gene can be isolated on the genome and does not have to be part of a cluster of genes to be considered for system's assessment (*default false*).
- **multi_system**: a *boolean*. If a gene has the feature "multi_system" (value set to "1", "true" or "True"), it means that it can be used to fill multiple system occurrences (from a same model) - and thus be considered as part of several systems (*default false*).

Fig. 1: How *loner* works.

A) The *cluster 1* can be filled up with the loner *D50* to reach the quorum defined in *model A* and form a system occurrence. **B)** There are 2 clusters and 2 loners (*D50* and *D60*) and *msf* cannot assign which loner goes to which cluster. So *msf* picks the best loner (based on score) and sets the others as “counterpart”. 2 system occurrences are created with the best loner. The user has to choose which loner hit can be assigned to which cluster. All loners found in the best solution are reported in *best_solution_loners.tsv* file. **C)** There are 2 clusters but only 1 loner. *msf* cannot decide to which cluster assign the loner. So the 2 system occurrences are proposed to the user in the output and a warning is raised to indicate the user should pick one. **D)** There are 2 clusters with one loner, but this loner is also *multi_system*. So the 2 clusters can be filled up with the loner.

- **multi_model**: a *boolean*. If a gene has the feature “multi_model” (value set to “1”, “true” or “True”), it means that two systems from different models can coexist in the best solution (they are said “compatible”) even if they share a component. The gene must be tagged as *multi_model* in both model definitions.

Fig. 2: How *multi_system* works.

A) The hit encoding for gene D in position 13 belongs to the system 1 (encoding model A). So it is used to fill up some other cluster, for instance cluster 2, which lacks this functionality. The cluster 2 then also fulfil the requirement of a system. **B)** The hit encoding for gene D in position 13 does not belong to a system. It cannot be used to fill up other clusters. In this example there is no system that satisfies the rules of model A. **C)** The gene D is present in the definition of model A and B. The hit encoding for gene D in position 13 belongs to the system 1 (encoding model A). It cannot be used to fill up the cluster 2 which codes for model B.

- **inter_gene_max_space**: an *integer* that defines gene-wise value of system's "inter_gene_max_space" parameter (see above). It supersedes the system-wise parameter to give the gene a specific co-localization parameter.

The element "gene" may have one "exchangeables" child element:

- The element "exchangeables" can contain one or more elements "gene".

For a Gene to have "exchangeables" Genes listed, means that this Gene can be replaced *in the quorum* by the listed child Genes.

Note

If the attributes *inter_gene_max_space*, *loner*, *multi_model*, *multi_system* are not specified for the exchangeable genes, then they inherit the values from the reference gene. Below some examples of attributes inheritance.

```
<gene name="A" presence="mandatory" multi_model="True">
  <exchangeables>
    <gene name="B" />
    <gene name="C" />
  </exchangeables>
</gene>
```

In the snippet code above, the genes A/B/C are *multi_model* but not *loner* or *multi_system*.

```
<gene name="A" presence="mandatory">
  <exchangeables>
    <gene name="B" multi_model="True"/>
    <gene name="C" />
  </exchangeables>
</gene>
```

In the snippet code above, The gene B is *multi_model* but not A and C.

```
<gene name="A" presence="mandatory" loner="True" multi_system="True">
  <exchangeables>
    <gene name="B" />
    <gene name="C" multi_system="False"/>
  </exchangeables>
</gene>
```

In the snippet code above,

- The genes A/B/C are *loner*
- The genes A and B are *multi_system*, but **not** C.

```
<gene name="A" presence="mandatory" inter_gene_max_space="10">
  <exchangeables>
    <gene name="B" inter_gene_max_space="5"/>
    <gene name="C" />
  </exchangeables>
</gene>
```

In the snippet code above, The genes A and C have an *inter_gene_max_space* = 10 whereas its value is 5 for the gene B .

Warning

The *presence* attribute is inevitably the same for the exchangeable genes than the reference gene.

Note

If not specified by the user, several features will have their values assigned **by default**:

- the **genomic architecture** of the System being searched will consist in a **single locus**. If a System may be made of Genes from multiple loci, consider setting the *multi_loci* parameter to *True*.
- the **quorum parameters** *min_mandatory_genes_required* and *min_genes_required* will be set to the number of mandatory Genes listed - the *accessory* Genes being deemed not required to infer a complete System.

Example of a macsy-model definition in XML (more examples in our *gallery of examples*):

```
<model inter_gene_max_space="5" vers="2.0">
  <gene name="gspD" presence="mandatory">
    <exchangeables>
```

(continues on next page)

(continued from previous page)

```

    <gene name="sctC"/>
  </exchangeables>
</gene>
<gene name="sctN_FLG" presence="mandatory" loner="1">
  <exchangeables>
    <gene name="gspE"/>
    <gene name="pilT"/>
  </exchangeables>
</gene>
<gene name="sctV_FLG" presence="mandatory"/>
<gene name="flp" presence="accessory"/>
</model>

```

In this example, the described System consists of three mandatory and one accessory components:

- Two components, the Gene “GspD” and the Gene “sctN_FLG” can respectively be replaced by sctC, and gspE and pilT genes in the quorum.
- To be considered as part of such System, the components should be co-localized in loci (Clusters of Genes), which in this case would amount to being located from each other at a distance of 5-Genes maximum, except for the Gene “sctN_FLG” that is allowed to be located “alone” in the genome being investigated, by a *loner* parameter being set to True. As the *multi_loci* parameter is not set, by default the System should be made of a single locus (Cluster of co-localized Genes - except for the ones listed as *loners*).
- To be considered a complete System, the quorum of Genes should be reached. In this case, the *min_genes_required* and *min_mandatory_genes_required* are not specified and therefore assigned to their default values: *min_mandatory_genes_required* is set to the number of mandatory Genes listed as well as the *min_genes_required* parameter (see above).

Warning

- a gene is identified by its name.
- this name is case sensitive.
- this name must be unique inside a family of models.
- a HMM profile with a gene-based name must exist in the *profiles* directory of the macy-model package (see *below*).

Providing HMM profiles

For each gene mentioned in each model you have to provide a **HMM profile** to enable the similarity search of this gene. The HMM profile must have been created by the user from a curated multiple sequence alignment with the *hmmbuild* program from the [HMMER package](#), or can have been obtained from HMM profiles’ databases such as [TIGRFAM](#) or [PFAM](#).

This profile *MUST* have the same name as the name of the gene mentioned in the definition. For instance, a component named “GeneA” in the macy-model would correspond by default to a HMM profile “GeneA.hmm” enclosed in the macy-model package. The names are **case-sensitive**. All HMM profiles must be placed in the *profiles* directory of the macy-model package.

Warning

- MacSyFinder does not support several profiles per file. Each hmm file **must** contains only one profile.

Note

For a detailed tutorial on how to define your mcsy-model's features, parameters and HMM profiles, you can have a look at our cookbook in [this book chapter](#).

Helper Tool

msf_profile

To help develop new models we provide the tool *msf_profile* which is to be used as post treatment.

It is ran over a previous macsyfinder analysis:

- it extracts from raw HMMER output files the hits and computes the profile coverage for each of them.
- it enables to filter the hits in a user-defined manner, to test other values of filtering parameters than those used with the MacSyFinder run.
- it writes down the results in a file in *tsv* format *hmm_coverage.tsv*.

```
usage: msf_profile [-h] [--coverage-profile COVERAGE_PROFILE] [--i-eval-sel I_EVALU_
  SEL] [--best-hits {score,i_eval,profile_coverage}]
                  [-p PATTERN] [-o OUT] [--index-dir INDEX_DIR] [-f] [-V] [-v] [--mute]
                  previous_run
```

[illegible]

msf_profile - Profile Helper Tool

```
positional arguments:
```

```
previous_run      The path to a macsyfinder results directory.
```

```
options:
```

`-h, --help` show this help message and exit

```
--coverage-profile COVERAGE_PROFILE
```

```
Minimal profile coverage required for the hit alignment with the
profile to allow the hit selection for systems
detection. (default no threshold)
```

```
--i-evaluate-sel I_EVALUATE_SEL
```

Maximal independent e-value for Hmmer hits to be selected for systems detection. (default: no selection based on

(continues on next page)

(continued from previous page)

```

                                i-evalue)
--best-hits {score,i_eval,profile_coverage}
                                If several hits match the same replicon, same gene. Select only
↪ the best one (based on best 'score' or 'i_evalue'
                                or 'profile_coverage')
-p, --pattern PATTERN
                                pattern to filter the hmm files to analyse.
-o, --out OUT
                                the path to a file to write results.
--index-dir INDEX_DIR
                                Specifies the path to a directory to store/read the sequence
↪ index when the sequence-db dir is not writable.
-f, --force
                                force to write output even the file already exists (overwrite
↪ it).
-V, --version
                                show program's version number and exit
-v, --verbosity
                                Increases the verbosity level. There are 4 levels: Error
↪ messages (default), Warning (-v), Info (-vv) and
                                Debug.(-vvv)
--mute
                                Mute the log on stdout. (continue to log on macsyfinder.log)
↪ (default: False)

```

For more details, visit the MacSyLib website and see the MacSyLib documentation.

For instance:

```
>msf_profile macsyfinder-2021XXXX_XX-XX-XX
```

will analyse the HMMER raw outputs stored in *macsyfinder-2021XXXX_XX-XX-XX/hmm_results* directory and the results will be stored in *macsyfinder-2021XXXX_XX-XX-XX/hmm_coverage.tsv* file

Setting filtering parameters

This helper tool is designed to help the user test the relevance of the HMM profiles used, what filtering parameters for HMMER to be used, and understand why some components might be unexpectedly missing from the MacSyFinder results. This can thus help to improve the models - for instance for the genomic location parameters (is a component not found cause it should be listed as a *loner*?).

Therefore by default, the filtering parameters are very loose so that most hits found with HMMER will be reported, even the weakest ones.

However, it is possible to filter hits to be extracted based on the profile coverage with *-coverage-profile* or the i-evalue (*-i-evalue-sel*) to be a bit more stringent.

Also, it is possible to use the *-best-hits* in order to report only the best hit for a given protein sequence when several profiles were matching hit.

Using patterns with “-pattern”

If in *previous_run/hmm_results* you have the following files:

```

previous_run/hmm_results/Archaeal-T4P_arCOG11238.search_hmm.out
previous_run/hmm_results/Archaeal-T4P_arCOG11520.search_hmm.out
previous_run/hmm_results/Archaeal-T4P_arCOG11777.search_hmm.out
previous_run/hmm_results/Archaeal-T4P_arCOG11778.search_hmm.out
previous_run/hmm_results/Archaeal-T4P_arCOG11936.search_hmm.out

```

(continues on next page)

(continued from previous page)

```
previous_run/hmmer_results/Archaeal-T4P_arCOG14515.search_hmm.out
previous_run/hmmer_results/ComM_comC.search_hmm.out
previous_run/hmmer_results/ComM_comEB.search_hmm.out
previous_run/hmmer_results/ComM_comEC.search_hmm.out
previous_run/hmmer_results/ComM_comGA.search_hmm.out
previous_run/hmmer_results/ComM_comGB.search_hmm.out
previous_run/hmmer_results/ComM_comGC.search_hmm.out
previous_run/hmmer_results/ComM_comGD.search_hmm.out
previous_run/hmmer_results/ComM_comGE.search_hmm.out
previous_run/hmmer_results/MSH_mshA.search_hmm.out
previous_run/hmmer_results/MSH_mshB.search_hmm.out
previous_run/hmmer_results/MSH_mshC.search_hmm.out
```

But you are interested only in ComM family genes, you can specify the option `--pattern 'ComM*'` For instance:

```
>msf_profile --pattern 'ComM*' macyfinder-2021XXXX_XX-XX-XX
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comB.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comC.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comEA.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comEB.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comEC.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comGA.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comGB.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comGC.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comGD.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comGE.search_hmm.out
found 79 hits
result is in 'macyfinder-2021XXXX_XX-XX-XX/hmm_coverage.tsv'
```

Note

The patterns available are the *glob* patterns (the jokers usable with unix *ls* command)

```
>msf_profile --pattern 'ComM_com?C' -f macyfinder-2021XXXX_XX-XX-XX
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comEC.search_hmm.out
parsing macyfinder-2021XXXX_XX-XX-XX/hmmer_results/ComM_comGC.search_hmm.out
found 16 hits
result is in 'macyfinder-2021XXXX_XX-XX-XX/hmm_coverage.tsv'
```

A useful example for modellers?

```
>msf_profile --best-hits i_eval --i-evalue-sel 0.001 --coverage-profile 0.5 -o msf_GCF_
↪003149495.1_ASM314949v1_tff-sf/hmm_coverage_best-hits_ieval_default_filter_MSF.tsv msf_
↪GCF_003149495.1_ASM314949v1_tff-sf
found 221 hits
result is in 'msf_GCF_003149495.1_ASM314949v1_tff-sf/hmm_coverage_best-hits_ieval_
↪default_filter_MSF.tsv'
```

This command line might be useful to macy-models modellers, as it consists in extracting all relevant hits that are used by the MacSyFinder engine to search systems, when using the default parameters:

- the proteins are assigned with their best hits (i-evalue based) when they match several profiles (`--best-hits i_eval`)

option)

- the default filtering parameters (i-evalue and profile coverage) are used (*-i-evalue-sel* and *-coverage-profile* options)

By using this command line that lists all hits available for MacSyFinder to search for systems, one could be interested in comparing this list to the list of hits that end in being assigned to systems (listed e.g. in *best_solution.tsv*). This can help to determine why a component is missing from a system: is it because there are no good hits for it, or is it because it does not comply to the co-localization rules defined in the systems' model?

Parsing msf_profile outputs

The *msf_profile* output is a tabulated separated values (*.tsv*) files. The first lines which are comments (starting with '#') display the tool version and the complete command line used. Then follow the results. The first line of results is a header line.

```
# msf_profile 2.1.5
# msf_profile --pattern ComM* --coverage-profile 0.5 macsyfinder-20201202_15-17-46/
hit_id replicon_name position_hit hit_sequence_length gene_name i_eval
↪score profile_coverage sequence_coverage begin end
GCF_000006745_021980 GCF_000006745 2198 291 ComM_comC 2.500e-40
↪136.400 0.942 0.708 62 267
GCF_000006745_007650 GCF_000006745 765 253 ComM_comC 9.600e-31
↪105.100 0.937 0.798 43 244
...
```

Note

This file can be easily parsed using the Python *pandas* library.

```
import pandas as pd

systems = pd.read_csv("path/to/hmm_coverage.tsv", sep='\t', comment='#')
```

Warning

The *msf_profile* tool is not compliant with results produced with *macsyfinder v1*. If you get *Cannot find models in conf file XXX*. May be these results have been generated with an old version of *macsyfinder*. Check the configuration file, if *[models]* section contains *models_1 = XXX YYY* remove the *_1* from models *models = XXX YYY*

Publishing/sharing models

Writing your own macsy-model package

The whole package structure and the corresponding files are described in the section *Structure of a macsy-model package*. It requires five different types of files to be complete:

- a *metadata.yml* file (mandatory)
- a *README.md* file (mandatory)

- a *LICENSE* file (optional but **HIGHLY** recommended)
- a *model_conf.xml* file (optional)
- macsy-models definition(s) within a *definitions* folder (mandatory)
- HMM profiles within a *profiles* folder (mandatory)

You can create a template for your package by using *msf_data init*. It will create for you:

- the git repository with the data package with the right structure.
- a template of *metadata.yaml*.
- a template of *README.md* file.
- a generic *model_conf.xml* file.
- a *LICENSE* file if *-license* option is set.
- a *COPYRIGHT* file if *-holders* option is set.
- a directory *definitions* with an example of model definition (*model_example.xml* to remove before publishing).
- a directory *profiles* where to put the hmm profiles corresponding to the models genes.

Sharing your models

If you want to share your models you can create a *macsy-model package* in your github repository. Several steps are needed to publish your model:

1. Check the **validity** of your package with the *msf_data check* command. You have to run it from within the folder containing your package files. It will report:
 - everything is clear: *msf_data* displays the next step to take to publish the package
 - warning: it means that the package could be improved.

It is better to fix it if you can, but you can also proceed to *Step 2*

 - error: the package is not ready to be published as is. You have to fix the errors before you go to *Step 2*.
2. Create a **tag**, and submit a **pull request** to the <https://github.com/macsy-models> organization. This step is **very important**: without a tag, there is no package. *msf_data check* only tagged packages. It is **Mandatory** to follow a versioning scheme described here:
 - <https://www.python.org/dev/peps/pep-0440/#public-version-identifiers>
 - <https://the-hitchhikers-guide-to-packaging.readthedocs.io/en/latest/specification.html#standard-versioning-schemes>

Important

If your package is in version *2.0.1* the tag must be *2.0.1*. The version or tag must **NOT** start with letter as *v2.0.1* or *my_package-2.0.1*.

Warning

To avoid making an inconsistent model visible by *msf_data install/search* (by pushing a tag), a pre-push hook has been setup in the git repository by *msf_data init* command. If you do not used *msf_data init* to create the model, It is a good idea to set up the hook by yourself.

Check that the hook is well named pre-push and it is executable (*chmod 755 .git/hooks/pre-push*) This script run *msf_data check* if you push a tag and it prevent the push if some error are found.

```
#!/bin/sh

# An example hook script to verify what is about to be pushed. Called by "git
# push" after it has checked the remote status, but before anything has been
# pushed. If this script exits with a non-zero status nothing will be pushed.
#
# This hook is called with the following parameters:
#
# $1 -- Name of the remote to which the push is being done
# $2 -- URL to which the push is being done
#
# If pushing without using a named remote those arguments will be equal.
#
# Information about the commits which are being pushed is supplied as lines to
# the standard input in the form:
#
# <local ref> <local oid> <remote ref> <remote oid>
#
# This script check if you push a tag
# if yes check if the tag match to the version decalred in metadata.yml
# if yes it prevents the push until the tag and the version match
#
# This script is widely inspired from https://gist.github.com/farseerfc/
→0729c08cd7c82b07000f20105f733b17

remote="$1"
url="$2"

tagref=$(grep -Po 'refs/tags/([^\ ]*)' </dev/stdin | head -n1 | cut -c11- | tr -
→d '[:space:]')

if [[ "$tagref" == "" ]]; then
    ## NOT pushing tag , exit normally
    exit 0
fi

macsydata check
returncode=$?

if [ $returncode -ne 0 ]; then
    Red='\e[1;31m'
    Green='\e[1;32m'
    Yello='\e[1;33m'
    Clear='\e[0m'
    echo "${Green}To fix errors:${Clear}"
    echo "${Red} 1. remove tag:${Clear} git tag -d ${tagref}"
    echo "${Yello} 2. fix errors above ${Clear}"
    echo "${Yello} 3. run 'macsydata check' until everything is fixed ${Clear}"
    echo "${Green} 4. commit your fix:${Clear} git add / git commit "
    echo "${Green} 5. tag again:${Clear} git tag -a ${tagref}"
    echo "${Green} 6. and push:${Clear} git push ${remote} ${tagref}"
```

```
fi
exit $returncode
pre-push .
```

3. When your pull request (PR) is accepted, the model package becomes automatically available to the community through the *msf_data* tool.

If you don't want to submit a PR you can provide the tag release tarball (tar.gz) as is to your collaborators. This archive will also be usable with the *msf_data* tool.

Note

msf_data check checks the syntax of the package, but it does not publish anything. It just warns you if something is wrong with the package. Every model provider should check its own package before publishing it. The package publication is done by the *git push* and the *pull request*.

Examples of *msf_data check* outputs:

Your package is syntactically correct:

```
msf_data check tests/data/models/test_model_package/
Checking 'test_model_package' package structure
Checking 'test_model_package' metadata_path
Checking 'test_model_package' Model definitions
Models Parsing
Definitions are consistent
Checking 'test_model_package' model configuration
There is no model configuration for package test_model_package.
If everyone were like you, I'd be out of business
To push the models in organization:
    cd tests/data/models/test_model_package
Transform the models into a git repository
    git init .
    git add .
    git commit -m 'initial commit'
add a remote repository to host the models
for instance if you want to add the models to 'macsy-models'
    git remote add origin https://github.com/macsy-models/
    git tag 1.0b2
    git push --tags
```

You received some warnings:

```
msf_data check tests/data/models/Model_w_conf/
Checking 'Model_w_conf' package structure
Checking 'Model_w_conf' metadata_path
Checking 'Model_w_conf' Model definitions
Models Parsing
Definitions are consistent
Checking 'Model_w_conf' model configuration
The package 'Model_w_conf' have not any LICENSE file. May be you have not right to use.
```

(continues on next page)

(continued from previous page)

`→it.`

The package 'Model_w_conf' have not any README file.

msf_data says: You're only giving me a partial QA payment?

I'll take it this time, but I'm not happy.

I'll be really happy, if you fix warnings above, before to publish these models.

You received some errors:

```
msf_data check tests/data/models/TFF-SF/
```

```
Checking 'TFF-SF' package structure
```

```
The package 'TFF-SF' have no 'metadata.yml'.
```

```
Please fix issues above, before publishing these models.
```

```
ValueError
```

Gallery of examples of MacSyFinder's models

Table of contents of the gallery

- *Getting started with a one-component system: the autotransporter T5SS*
- *A (not-so)-simple example: modelling the T1SS*
- *The case of T3SS and bacterial flagella, or how to distinguish homologous cellular machineries*

Here follows a “gallery” of MacSyFinder models we have developed over the years, attempting to describe the reasoning behind the modeling process.

These examples are extracted from published work, see the following references (they include more examples):

- [Abby et al. 2016, *Scientific Reports*](#), for the description of the T1SS, T3SS and T5aSS models (and way more models not discussed here).
- [Abby and Rocha 2012, *PLoS Genetics*](#), for the evolutionary study of the T3SS and the bacterial flagellum, and how were designed the corresponding profiles.
- [Denise et al. 2019, *PLoS Biology*](#), for the description of the T2SS and type IV-filament super-family models.

Getting started with a one-component system: the autotransporter T5SS

This case is rather straight-forward, as the detection of the autotransporter type V secretion system (T5aSS) relies solely on the detection of a single component. This system indeed encodes both a translocator (outer membrane, pore-forming domain) and a passenger domain (toxin or enzyme) on the same gene.

The translocator domain is the **evolutionarily conserved** part across T5aSS. This family of homologous proteins is gathered in the PFAM protein family [PF03797](#) of “Autotransporter” domains.

We thus downloaded the corresponding pre-computed HMM profile that we named “T5aSS_PF03797.hmm” to enable its search using sequence similarity.

We then wrote the corresponding MacSyFinder model in a file **T5aSS.xml**:

```
<model inter_gene_max_space="1" vers="2.0">
  <gene name="T5aSS_PF03797" presence="mandatory"/>
</model>
```

It can be noted that several features do not have to be defined if default values are relevant. In particular, in this example it is not needed to specify the quorum parameters: the default value for the minimal number of genes required to infer the presence of the T5aSS is by default the number of components listed in the definition of the system (1).

A (not-so-)simple example: modelling the T1SS

1. Identifying genetic components

The type I secretion system (T1SS) consists in three conserved components:

- an ABC transporter (ABC)
- a membrane-fusion protein (MFP)
- an outer membrane protein (OMF)

For their detection, we therefore need to provide HMM profiles for each component, for example: “abc.hmm”, “mfp.hmm” and “omf.hmm”. These can be specifically designed, or taken from HMM profiles databanks such as [PFAM](#), [TIGRFAM](#) or [SUPERFAMILY](#)..

Note

For suggestions on how to design specific HMM protein profiles, read our dedicated book chapter:

[Identification of Protein Secretion Systems in Bacterial Genomes Using MacSyFinder](#) by Sophie Abby and Eduardo Rocha, in *Methods in Molecular Biology* (2017).

2. Determining the role of the components

From literature, the three components listed above *must* be present to have a viable T1SS. Therefore, these are all deemed *mandatory* in the model of the T1SS.

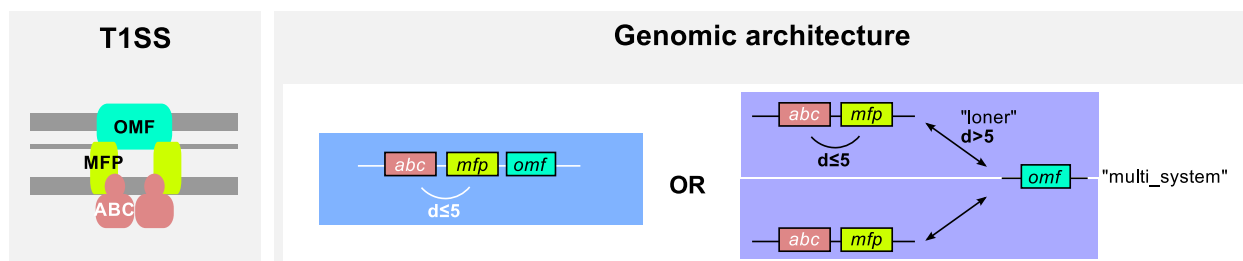
3. Describing their genetic architecture

According to the litterature, the genes encoding the three components listed above are generally found lying next to each other in genomes. Therefore, these are considered as “single-locus” system. In addition, there is the particular case of the OMF component. It can either be found:

- next to the two other components, as explained just below
- in some other cases, it can be involved in other cellular machineries functioning, and thus be encoded some place else that at the main T1SS’ locus (in this case, made of ABC+MFP).

Therefore, we can attribute the *loner* feature to the OMF component.

In addition to the latter exception described, it means that this OMF component can also be involved in the functioning of not a single, but several machineries at the same time. In practice, this would mean that two full sets of T1SS components can be inferred with a single OMF component found in the genome. This corresponds to the *multi-system* feature.



4. Writing down the model

Now that all elements of the model are listed, the model for the T1SS can be written using the dedicated MacSyFinder XML grammar:

```
<model inter_gene_max_space="5" min_mandatory_genes_required="3" min_genes_required="3"
↪vers="2.0">
  <gene name="T1SS_abc" presence="mandatory"/>
  <gene name="T1SS_mfp" presence="mandatory"/>
  <gene name="T1SS_omf" presence="mandatory" loner="1" multi_system="1"/>
</model>
```

The case of T3SS and bacterial flagella, or how to distinguish homologous cellular machineries

The type III secretion system (T3SS), involved in proteic effectors secretion into eukaryotic cells) and the bacterial flagellum (involved in motility) are evolutionarily related (Abby and Rocha 2012). This can make their annotation in genomes tricky, if only based on core components that can have homologs in both systems.

However, these machineries also have **specific core components**. With MacSyFinder and the *forbidden* feature for components, it is possible to model this, and create models for efficient discrimination between homologous machineries.

For a toy example on how to model similar yet distinct machineries, you can also have a look [here](#).

1. Identifying genetic components and determining their role

The T3SS is partly homologous to the bacterial flagellum: 8 of its 9 core components are homologous to core components of the flagellum. This is explained by the fact that the T3SS is evolutionarily derived from the flagellum (Abby and Rocha 2012). Yet, the T3SS is made of two dozens of components, and the flagellum, more than twice this number of components:

- The flagellum presents specific core components that have no counterpart in the T3SS.
- It is also the case of the T3SS, which has one specific core component: the secretin.

Solely based on the specificity of core components, it is possible to distinguish T3SS from flagella. This can be done by listing the **specific core components** of a given system as *mandatory* in the system, and as *forbidden* in the homologous system.

Then, HMM protein profiles can be specifically designed for these components, or can be retrieved from databases such as [PFAM](#), [TIGRFAM](#) or [SUPERFAMILY](#).

2. Dealing with components with varied evolutionary origins

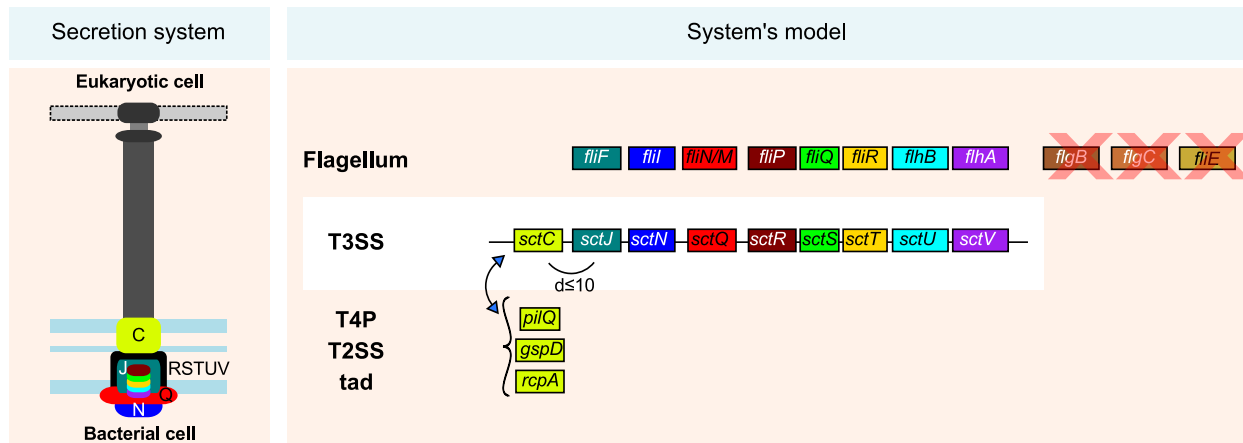
Another peculiarity of T3SS' evolutionary history consists in that of the secretin, which has been co-opted (acquired) at least three times independently along T3SS diversification: once from the T2SS, once from the Tad pilus, and once from the Type IVa pilus (Abby and Rocha 2012, Denise et al. 2019).

This means that sometimes, the T3SS secretin will have more sequence similarity for the secretins from these other machineries - and thus that the profile for the T3SS secretin might “miss” these components, whereas profiles for secretins from the T2SS, T4P or Tad might be more efficient to retrieve them.

Using the *exchangeables* feature, MacSyFinder enables to use different HMM protein profile to search for components that may fill a same function. Therefore, it is possible to list profiles of secretins from other machineries among the set of profiles to use to retrieve all T3SS potential secretins.

In the following drawing, a scheme of a T3SS is shown on the left, and the features listed above are shown on a scheme of the T3SS model, including forbidden components from the flagellum (red crosses), and exchangeable components

for the secretin “sctC”, depicted with yellow boxes (with the name of the secretin gene from the T4aP, T2SS and Tad pilus respectively). The *inter-gene-max-space* parameter - i.e., maximal number of components allowed between two systems’ components to consider them consecutive - is expressed with the “d” letter.



3. Describing the quorum, and genetic architecture of the systems

- T3SS and bacterial flagella are generally encoded on the form of multi-components loci in genomes. Given the fact that we designed HMM protein profiles only for the most conserved, core components of these machineries, and that it means that several systems’ components can intersperse between the core ones (remember, T3SS has around 25 components, and the flagellum >40), we set the *inter-gene-max-space* parameter (maximal number of components allowed between two systems’ components to consider them consecutive) to 10 in the case of the T3SS, and to 20 in the case of the flagellum.
- T3SS and bacterial flagella can be encoded by one, or multiple loci. We therefore use the *multi-loci* feature to describe their genetic architecture (set to “1”, meaning “True” in the models).

Note

For suggestions on how to set the quorum and genetic architecture parameters, read our dedicated book chapter:

Identification of Protein Secretion Systems in Bacterial Genomes Using MacSyFinder by Sophie Abby and Eduardo Rocha, in *Methods in Molecular Biology* (2017).

4. Writing down the models

Given all the features described above, here is the model of the T3SS:

T3SS.xml

```
<model inter_gene_max_space="10" min_mandatory_genes_required="7" min_genes_required="7"
multi_loci="1" vers="2.0">
  <gene name="T3SS_sctC" presence="mandatory">
    <exchangeables>
      <gene name="T2SS_gspD"/>
      <gene name="T4P_pilQ"/>
      <gene name="Tad_rcpA"/>
    </exchangeables>
  </gene>
  <gene name="T3SS_sctJ" presence="mandatory"/>
```

(continues on next page)

(continued from previous page)

```

<gene name="T3SS_sctN" presence="mandatory"/>
<gene name="T3SS_sctQ" presence="mandatory"/>
<gene name="T3SS_sctR" presence="mandatory"/>
<gene name="T3SS_sctS" presence="mandatory"/>
<gene name="T3SS_sctT" presence="mandatory"/>
<gene name="T3SS_sctU" presence="mandatory"/>
<gene name="T3SS_sctV" presence="mandatory"/>
<gene name="Flg_fliE" presence="forbidden"/>
<gene name="Flg_flgB" presence="forbidden"/>
<gene name="Flg_flgC" presence="forbidden"/>
</model>

```

And the model of the Flagellum:

Flagellum.xml

```

<model inter_gene_max_space="20" min_mandatory_genes_required="9" min_genes_required="10"
  multi_loci="1" vers="2.0">
  <gene name="Flg_sctJ_FLG" presence="mandatory"/>
  <gene name="Flg_sctN_FLG" presence="mandatory"/>
  <gene name="Flg_sctQ_FLG" presence="mandatory"/>
  <gene name="Flg_sctR_FLG" presence="mandatory"/>
  <gene name="Flg_sctS_FLG" presence="mandatory"/>
  <gene name="Flg_sctT_FLG" presence="mandatory"/>
  <gene name="Flg_sctU_FLG" presence="mandatory"/>
  <gene name="Flg_sctV_FLG" presence="mandatory"/>
  <gene name="Flg_flgB" presence="mandatory"/>
  <gene name="Flg_flgC" presence="mandatory"/>
  <gene name="Flg_fliE" presence="mandatory"/>
  <gene name="T3SS_sctC" presence="forbidden"/>
</model>

```

2.1.2 Carrying models from v1 to v2

Carrying models from v1 to v2

Models from v1 are not compatible straight away with v2. For those who had designed MacSyFinder's models for Version 1 and would like to carry them for Version 2, here are the changes to consider:

- the keyword "system" was changed: `<system>` ::arrow:: `<model>`
- the keyword `<system_ref>` was removed. For a given systems' package, each gene has to be defined only once in a macsy-model. There is no need anymore to reference which model it is from, when used as a component in another system's model.
- now the version of the macsy-models' type has to be documented as a feature of the "model" keyword, like this: `vers = "2.0"`
- the following keywords have been replaced (but see *below* for more details):
 - homologs => exchangeables
 - analogs => exchangeables

Note

“exchangeable” is not a feature anymore, but is replaced by the keyword “exchangeables”.

Note

These changes in the grammar used to specify model is also accompanied by a change on how to organize folders with models and profiles. In particular, the new file architecture enables an *easier shipping* of the developed macsy-models. See [here](#) for more details.

Here follow some examples of updates from v1 to v2.

1. A very simple model.

TISS.xml under v1:

```
<system inter_gene_max_space="5" min_mandatory_genes_required="3" min_genes_required="3">
  <gene name="T1SS_abc" presence="mandatory"/>
  <gene name="T1SS_mfp" presence="mandatory"/>
  <gene name="T1SS_omf" presence="mandatory" loner="1" multi_system="1"/>
</system>
```

TISS.xml under v2:

```
<model inter_gene_max_space="5" min_mandatory_genes_required="3" min_genes_required="3"
↪ vers = "2.0">
  <gene name="T1SS_abc" presence="mandatory"/>
  <gene name="T1SS_mfp" presence="mandatory"/>
  <gene name="T1SS_omf" presence="mandatory" loner="1" multi_system="1"/>
</model>
```

Note

In a nutshell, the minimal changes from v1 to v2 for a simple macsy-model listing components are the following:

- `<system> => <model>`
- `vers = "2.0"`

2. A model with homologs.

Tad.xml under v1:

```
<system inter_gene_max_space="5" min_mandatory_genes_required="4" min_genes_required="6"
↪ multi_loci="0">
  <gene name="Tad_rcpA" presence="mandatory">
    <homologs>
      <gene name="T2SS_gspD" system_ref="T2SS"/>
      <gene name="T4P_pilQ" system_ref="T4P"/>
      <gene name="T3SS_sctC" system_ref="T3SS"/>
    </homologs>
  </gene>
</system>
```

(continues on next page)

(continued from previous page)

```

</gene>
<gene name="Tad_tadA" presence="mandatory"/>
<gene name="Tad_tadB" presence="mandatory"/>
<gene name="Tad_tadC" presence="mandatory"/>
<gene name="Tad_tadV" presence="mandatory"/>
<gene name="Tad_tadZ" presence="mandatory"/>
<gene name="Tad_flp" presence="accessory"/>
<gene name="Tad_tadE" presence="accessory"/>
<gene name="Tad_tadF" presence="accessory"/>
</system>

```

Tad.xml under v2:

```

<model inter_gene_max_space="5" min_mandatory_genes_required="4" min_genes_required="6"
↪ multi_loci="0" vers="2.0">

  <gene name="Tad_rcpA" presence="mandatory"/>
  <gene name="Tad_tadA" presence="mandatory"/>
  <gene name="Tad_tadB" presence="mandatory"/>
  <gene name="Tad_tadC" presence="mandatory"/>
  <gene name="Tad_tadV" presence="mandatory"/>
  <gene name="Tad_tadZ" presence="mandatory"/>
  <gene name="Tad_flp" presence="accessory"/>
  <gene name="Tad_tadE" presence="accessory"/>
  <gene name="Tad_tadF" presence="accessory"/>

</model>

```

Note

The *homologs* and *analogs* keyword having disappeared, it is not necessary anymore to list homologous components (e.g., those that may match several HMM profiles during the sequence similarity search), unless they are *exchangeables*.

3. A model with exchangeable homologs.

T3SS.xml under v1:

```

<system inter_gene_max_space="10" min_mandatory_genes_required="7" min_genes_required="7"
↪ multi_loci="1">
  <gene name="T3SS_sctC" presence="mandatory" exchangeable="1">
    <homologs>
      <gene name="T2SS_gspD" system_ref="T2SS"/>
      <gene name="T4P_pilQ" system_ref="T4P"/>
      <gene name="Tad_rcpA" system_ref="Tad"/>
    </homologs>
  </gene>
  <gene name="T3SS_sctJ" presence="mandatory">
    <homologs>
      <gene name="Flg_sctJ_FLG" system_ref="Flagellum"/>
    </homologs>

```

(continues on next page)

(continued from previous page)

```

</gene>
<gene name="T3SS_sctN" presence="mandatory">
  <homologs>
    <gene name="Flg_sctN_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="T3SS_sctQ" presence="mandatory">
  <homologs>
    <gene name="Flg_sctQ_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="T3SS_sctR" presence="mandatory">
  <homologs>
    <gene name="Flg_sctR_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="T3SS_sctS" presence="mandatory">
  <homologs>
    <gene name="Flg_sctS_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="T3SS_sctT" presence="mandatory">
  <homologs>
    <gene name="Flg_sctT_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="T3SS_sctU" presence="mandatory">
  <homologs>
    <gene name="Flg_sctU_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="T3SS_sctV" presence="mandatory">
  <homologs>
    <gene name="Flg_sctV_FLG" system_ref="Flagellum"/>
  </homologs>
</gene>
<gene name="Flg_fliE" presence="forbidden" system_ref="Flagellum"/>
<gene name="Flg_flgB" presence="forbidden" system_ref="Flagellum"/>
<gene name="Flg_flgC" presence="forbidden" system_ref="Flagellum"/>
</system>

```

T3SS.xml under v2:

```

<model inter_gene_max_space="10" min_mandatory_genes_required="7" min_genes_required="7"
multi_loci="1" vers="2.0">
  <gene name="T3SS_sctC" presence="mandatory">
    <exchangeables>
      <gene name="T2SS_gspD"/>
      <gene name="T4P_pilQ"/>
      <gene name="Tad_rcpA"/>
    </exchangeables>
  </gene>

```

(continues on next page)

(continued from previous page)

```

<gene name="T3SS_sctJ" presence="mandatory"/>
<gene name="T3SS_sctN" presence="mandatory"/>
<gene name="T3SS_sctQ" presence="mandatory"/>
<gene name="T3SS_sctR" presence="mandatory"/>
<gene name="T3SS_sctS" presence="mandatory"/>
<gene name="T3SS_sctT" presence="mandatory"/>
<gene name="T3SS_sctU" presence="mandatory"/>
<gene name="T3SS_sctV" presence="mandatory"/>
<gene name="Flg_fliE" presence="forbidden"/>
<gene name="Flg_flgB" presence="forbidden"/>
<gene name="Flg_flgC" presence="forbidden"/>
</model>

```

Note

- As only the secretin component ‘T3SS_sctC’ was exchangeable in its role within T3SS with its homologs T2SS_gspD, T4P_pilQ and Tad_rcpA, these three components are now set as *exchangeables* (they can functionally *replace* the component ‘T3SS_sctC’), and all other *homologs* do not need to be listed anymore.
- The keyword *system_ref* is not needed anymore. Therefore, the **v2** definition of T3SS is way more compact than that for **v1**.

2.1.3 Frequently Asked Questions

Frequently Asked Questions

How to report an issue?

If you encounter a problem while running MacSyFinder, please submit an issue on the dedicated page of the [GitHub project](#)

To ensure we have all elements to help, please provide:

- a concise description of the issue
- the expected behavior VS observed one
- the exact command-line used
- the version of MacSyFinder used
- the exact error message, and if applicable, the *macsyfinder.log* and *macsyfinder.conf* files
- if applicable, an archive (or link to it) with the output files obtained
- if possible, the smallest dataset there is to reproduce the issue
- if applicable, this would also include the macsy-models (XML models plus HMM profiles) used (or precise version of the models if there are publicly available). Same as above, if possible, please provide the smallest set possible of models and HMM profiles.

All these will definitely help us to help you! ;-)

How to list several components or HMM profiles for a given function in the model?

MacSyFinder provides a framework to associate a component/function in the model of a system with the mean to search for it - a HMM profile.

In some cases, it is needed to list several possible components (i.e. HMM profiles) to assume a given function for the system to model. There can be several reasons for that:

- a biological reason (e.g., two components from two different gene families can assume a same role in the system)
- a methodological reason (it is not possible or difficult to provide a single HMM profile that covers the diversity of the components' sequences to be retrieved).

It is possible to list several possible components for a same role within the system's model using the *exchangeables* keyword.

See [here](#) for more details and examples.

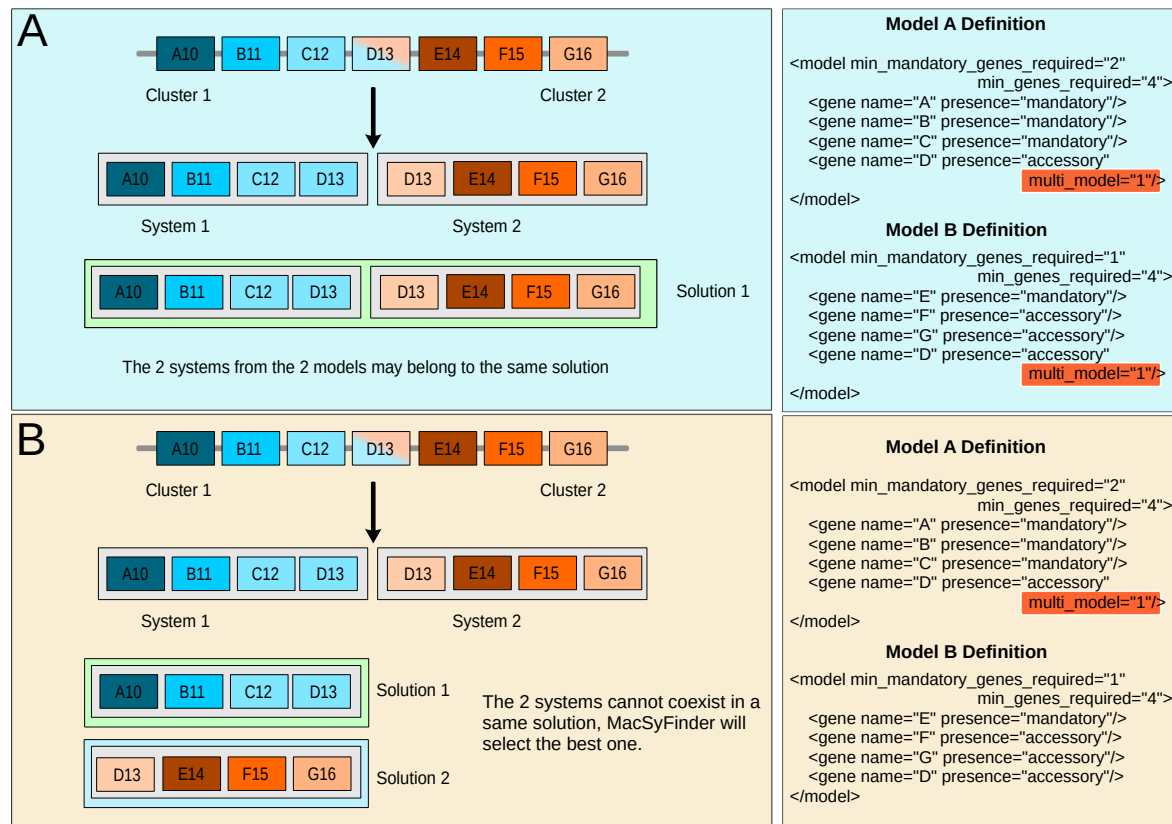


Fig. 3: How *multi_model* works.

The hit encoding for gene D in position 13 is part of 2 systems: one for Model A, one for Model B. **A**) In both model definitions the gene D is tagged as *multi_model*. So the 2 systems can coexist in a same solution (they are “compatible”). **B**) The gene D is tagged as *multi_model* **only** in model A definition. The 2 systems are not compatible. So *msf* build 2 solutions and choose the best one. It has to be noted that this behaviour would actually be the same if gene D was not declared *multi_model* in either definitions.

DEVELOPER GUIDE

3.1 Developer Guide

3.1.1 Installation

MacSyFinder works with models for macromolecular systems that are not shipped with it, you have to install them separately. See the *msf_data* section below. We also provide container so you can use macsyfinder directly.

MacSyFinder dependencies

Python version ≥ 3.10 is required to run MacSyFinder: <https://docs.python.org/3.10/index.html>

MacSyFinder has one program dependency:

- the *Hmmer* program, version 3.1 or greater (<http://hmmer.org/>).

The *hmmsearch* program should be installed (*e.g.*, in the PATH) in order to use MacSyFinder. Otherwise, the paths to this executable must be specified in the command-line: see the *command-line options*.

MacSyFinder also relies on some Python library dependencies:

- MacSyLib
- colorlog
- colorama
- pyyaml
- packaging
- networkx
- pandas
- GitPython
- sphinx
- sphinx_rtd_theme
- sphinx-autodoc-typehints
- sphinxcontrib-svg2pdfconverter
- coverage
- build
- ruff
- pre-commit

These dependencies will be automatically retrieved and installed when using *pip* for installation (see below).

MacSyFinder Installation and testing procedures

Installation steps:

Make sure every required dependency/software is present.

By default MacSyFinder will try to use *hmmsearch* in your PATH. If *hmmsearch* is not in the PATH, you have to set the absolute path to *hmmsearch* in a *configuration file* or in the *command-line* upon execution. If the tools are not in the path, some test will be skipped and a warning will be raised.

installation in a virtualenv

```
# create a new virtualenv
python3 -m venv macsyfinder
# activate it
cd macsyfinder
source bin/activate
# clone/install the project in editable mode
git clone
cd macsyfinder
python3 -m pip install -e .[dev]
# install tools to ensure coding style
pre-commit install
```

To exit the virtualenv just execute the *deactivate* command.

```
source macsyfinder/bin/activate
```

Then run *macsyfinder* or *msf_data*.

Note

from 2.1.4 version, *MacSyFinder* has adopted *ruff* as linter and *pre-commit* to ensure the coding style. please read *CONTRIBUTING.md* guide lines.

Testing

MacSyFinder project use *unittest* framework (included in the standard library) to test the code.

All tests stuff is in *tests* directory.

- The data directory contains data needed by the tests
- in the *__init__.py* file a *MacsyTest* class is defined and should be the base of all testcase use in the project
- each *test_*.py* represent a file containing unit or functional tests

To run all the tests (in the virtualenv)

```
python -m unittest discover
```

To increase verbosity of output


```
python -m unittest discover -vv
```

```
...
test_check_choice (tests.test_macsyconfig.TestMacsyconfig.test_check_choice) ... ok
test_check_dir (tests.test_macsyconfig.TestMacsyconfig.test_check_dir) ... ok
test_check_exe (tests.test_macsyconfig.TestMacsyconfig.test_check_exe) ... ok
test_check_file (tests.test_macsyconfig.TestMacsyconfig.test_check_file) ... ok
test_check_float (tests.test_macsyconfig.TestMacsyconfig.test_check_float) ... ok
test_check_positive_int (tests.test_macsyconfig.TestMacsyconfig.test_check_positive_int) ...
↳... ok
test_check_str (tests.test_macsyconfig.TestMacsyconfig.test_check_str) ... ok
test_functional_dark_theme (tests.test_macsyconfig.TestMacsyconfig.test_functional_dark_
↳theme) ... ok
test_functional_file_already_exists (tests.test_macsyconfig.TestMacsyconfig.test_
↳functional_file_already_exists) ... ok
test_parse_args (tests.test_macsyconfig.TestMacsyconfig.test_parse_args) ... ok
test_set_base_options (tests.test_macsyconfig.TestMacsyconfig.test_set_base_options) ...
↳ok
test_set_general_options (tests.test_macsyconfig.TestMacsyconfig.test_set_general_
↳options) ... ok
test_set_hmmmer_options (tests.test_macsyconfig.TestMacsyconfig.test_set_hmmmer_options) ..
↳. ok
test_set_path_options (tests.test_macsyconfig.TestMacsyconfig.test_set_path_options) ...
↳ok
test_set_path_options_with_system_models_dir (tests.test_macsyconfig.TestMacsyconfig.
↳test_set_path_options_with_system_models_dir) ... ok
test_set_score_options (tests.test_macsyconfig.TestMacsyconfig.test_set_score_options) ..
↳. ok
test_set_section (tests.test_macsyconfig.TestMacsyconfig.test_set_section) ... ok
test_set_section_use_defaults (tests.test_macsyconfig.TestMacsyconfig.test_set_section_
↳use_defaults) ... ok
test_db_type_set_to_gembase (tests.test_macsyfinder.TestMacsyfinder.test_db_type_set_to_
↳gembase) ... ok
test_get_version_message (tests.test_macsyfinder.TestMacsyfinder.test_get_version_
↳message) ... ok
test_list_models (tests.test_macsyfinder.TestMacsyfinder.test_list_models) ... ok
test_list_models_no_permissions (tests.test_macsyfinder.TestMacsyfinder.test_list_models_
↳no_permissions) ... ok
test_parse_args (tests.test_macsyfinder.TestMacsyfinder.test_parse_args) ... ok
test_search_systems (tests.test_macsyfinder.TestMacsyfinder.test_search_systems) ... ok
test_search_systems_unordered (tests.test_macsyfinder.TestMacsyfinder.test_search_
↳systems_unordered) ... ok
test_cmde (tests.test_scripts.Test_macsyfinder.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_config.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_data.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_merge.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_profile.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_split.test_cmde) ... ok
```

```
-----
Ran 85 tests in 8.339s
```

```
OK
```

The tests must be in python file (.py) starting with with *test_*. It's possible to specify one or several test files, one module, or one class in a module or a method in a Test class.

Test the *test_macsyfinder* module

```
python -m unittest -vv tests.test_macsyfinder
```

```
test_db_type_set_to_gembase (tests.test_macsyfinder.TestMacsyfinder.test_db_type_set_to_
↳gembase) ... ok
test_get_version_message (tests.test_macsyfinder.TestMacsyfinder.test_get_version_
↳message) ... ok
test_list_models (tests.test_macsyfinder.TestMacsyfinder.test_list_models) ... ok
test_list_models_no_permissions (tests.test_macsyfinder.TestMacsyfinder.test_list_models_
↳no_permissions) ... ok
test_parse_args (tests.test_macsyfinder.TestMacsyfinder.test_parse_args) ... ok
test_search_systems (tests.test_macsyfinder.TestMacsyfinder.test_search_systems) ... ok
test_search_systems_unordered (tests.test_macsyfinder.TestMacsyfinder.test_search_
↳systems_unordered) ... ok
```

```
-----
Ran 7 tests in 1.773s
```

```
OK
```

Test only the class *TestMacsyfinder*

```
python -m unittest -vv tests.test_macsyfinder.TestMacsyfinder
```

```
test_db_type_set_to_gembase (tests.test_macsyfinder.TestMacsyfinder.test_db_type_set_to_
↳gembase) ... ok
test_get_version_message (tests.test_macsyfinder.TestMacsyfinder.test_get_version_
↳message) ... ok
test_list_models (tests.test_macsyfinder.TestMacsyfinder.test_list_models) ... ok
test_list_models_no_permissions (tests.test_macsyfinder.TestMacsyfinder.test_list_models_
↳no_permissions) ... ok
test_parse_args (tests.test_macsyfinder.TestMacsyfinder.test_parse_args) ... ok
test_search_systems (tests.test_macsyfinder.TestMacsyfinder.test_search_systems) ... ok
test_search_systems_unordered (tests.test_macsyfinder.TestMacsyfinder.test_search_
↳systems_unordered) ... ok
```

```
-----
Ran 7 tests in 1.862s
```

```
OK
```

Test only the method *test_list_models_no_permissions* from the test Class *TestMacsyfinder* in module *test_macsyfinder*

```
python -m unittest -vv tests.test_macsyfinder.TestMacsyfinder.test_list_models_no_
↳permissions
```

```
test_list_models_no_permissions (tests.test_macsyfinder.TestMacsyfinder.test_list_models_
↳no_permissions) ... ok
```

(continues on next page)

(continued from previous page)

```
-----
Ran 1 test in 0.000s
```

```
OK
```

Coverage

To compute the tests coverage, we use the [coverage](#) package. The package is automatically installed if you have installed *macsyfinder* with the *dev* target see [installation](#). The coverage package is setup in the *pyproject.toml* configuration file.

To compute the coverage

```
coverage run
```

```
...
test_set_score_options (tests.test_macsyconfig.TestMacsyconfig.test_set_score_options) ..
↪. ok
test_set_section (tests.test_macsyconfig.TestMacsyconfig.test_set_section) ... ok
test_set_section_use_defaults (tests.test_macsyconfig.TestMacsyconfig.test_set_section_
↪use_defaults) ... ok
test_db_type_set_to_gembase (tests.test_macsyfinder.TestMacsyfinder.test_db_type_set_to_
↪gembase) ... ok
test_get_version_message (tests.test_macsyfinder.TestMacsyfinder.test_get_version_
↪message) ... ok
test_list_models (tests.test_macsyfinder.TestMacsyfinder.test_list_models) ... ok
test_list_models_no_permissions (tests.test_macsyfinder.TestMacsyfinder.test_list_models_
↪no_permissions) ... ok
test_parse_args (tests.test_macsyfinder.TestMacsyfinder.test_parse_args) ... ok
test_search_systems (tests.test_macsyfinder.TestMacsyfinder.test_search_systems) ... ok
test_search_systems_unordered (tests.test_macsyfinder.TestMacsyfinder.test_search_
↪systems_unordered) ... ok
test_cmde (tests.test_scripts.Test_macsyfinder.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_config.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_data.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_merge.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_profile.test_cmde) ... ok
test_cmde (tests.test_scripts.Test_msf_split.test_cmde) ... ok

-----
Ran 85 tests in 8.339s

OK
```

Then display a report

```
coverage report
```

Name	Stmts	Miss	Branch	BrPart	Cover
src/macsyfinder/__init__.py	23	2	0	0	91%
src/macsyfinder/io.py	11	0	4	0	100%

(continues on next page)

(continued from previous page)

src/macsyfinder/scripts/macsy_gembase_split.py	95	2	22	3	96%
src/macsyfinder/scripts/macsy_merge_results.py	203	15	72	8	90%
src/macsyfinder/scripts/macsyconfig.py	245	5	88	4	97%
src/macsyfinder/scripts/msf.py	237	8	48	4	96%

TOTAL

or generate a html report

coverage html

Wrote HTML report to htmlcov/index.html

The results are in the *htmlcov* directory. With you favourite web browser, open the *index.html* file. for more options please refer to the [coverage documentation](#).

3.1.2 MacSyFinder implementation overview

MacSyFinder is implemented with an object-oriented architecture. Below a short glossary to fix the vocabulary used in MacSyFinder

Cluster

Is a “contiguous” set of hits. two hits are considered contiguous if the number of genes between the 2 genes matching the 2 hits in the replicon is lesser than inter-genes-max-space.

Model

Is a formal description of a macromolecular system. Is composed of a definition and a list of profiles. at each gene of the Model must correspond a profile

Model family

A set of models, on the same topic. It is composed of several definitions which can be sorted in hierachical structure and profiles. A profile is a hmm profile file.

ModelDefinition

Is a definition of model, it's serialize as a xml file

Solution

It's a systems combination for one replicon. The best solution for a replicon, is the combination of all systems found in this replicon which maximize the score.

System

It's an occurrence of a specific Model on a replicon. Basically, it's a cluster or set of clusters which satisfy the Model quorum.

MacSyFinder project structure

A brief overview of the files and directory constituting the MacSyFinder project

doc

The project is documented using sphinx. All sources files needed to generate this documentation is in the directory *doc*

macsyfinder

This the MacSyFinder python library. Inside macsyfinder there is a subdirectory *scripts* which are the entry points for *macsyfinder*, *msf_data*, *msf_profile*, ...

tests

The code is tested using *unittests*. In *tests* the directory *data* contains all data needed to perform the tests.

utils

Contains a binary *setsid* needed macsyfinder to parallelize some steps. Usually *setsid* is provided by the system, but some macOS version does not provide it.

CITATION.yml

A file indicating how to cite macsyfinder in yaml format.

CONTRIBUTORS

A file containing the list of code contributors.

CONTRIBUTING

A guide on how to contribute to the project.

COPYRIGHT

The macsyfinder copyrights.

COPYING

The licencing. MacSyFinder is released under GPLv3.

README.md

Brief information about the project.

setup.py

The installation recipe specific to embedded doc in distrib.

pyproject.toml

a configuration file for packaging-related tools (as well as other tools).

MacSyFinder architecture overview

Starting with 2.1.5 version MacSyFinder is built on top of *MacSyLib* for the internal read [MacSyLib documentation](#)

3.1.3 MacSyFinder API documentation**io**

Defined a specific header that is passed to macsylib.io functions.

io API reference**outfile_header**

`macsyfinder.io.outfile_header(models_fam_name: str, models_version: str, skipped_replicons: list[str] | None = None) → str`

Returns

The first lines of each result file

scripts

There are 4 entry points.

- `macsyfinder`: which is the main script
- `macsydata`: which allow to manage the models (command name `msf_data`)
- `macsyconfig`: an interactive conversational utility to generate macsyfinder configuration file (command name `msf_config`)
- `macsyprofile`: an utility dedicated to modelers which gather information about hmmer output (command name `msf_profile`)

API reference

macsyfinder

Main entrypoint to macsyfinder

`macsyfinder.scripts.msf.alarm_handler(signum: Signals, frame) → None`

Handle signal alarm flush loggers :param signum: :param frame: :raise: Timeout

`macsyfinder.scripts.msf.list_models(args: Namespace) → str`

Parameters

args – The command line argument once parsed

Returns

a string representation of all models and submodels installed.

`macsyfinder.scripts.msf.main(args: list[str] | None = None, loglevel: Literal['NOTSET', 'DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL'] | int | None = None)`

main entry point to MacSyFinder do some check before to launch `main_search_systems()` which is the real function that perform a search

Parameters

- **args** – the arguments passed on the command line without the program name
- **loglevel** – the output verbosity

`macsyfinder.scripts.msf.parse_args(args: list[str]) → tuple[ArgumentParser, Namespace]`

Parameters

args – The arguments provided on the command line

Returns

The arguments parsed

macsydata

wrapper to `macsylib.macsydata`

macsyconfig

Entrypoint for `macsyconfig` command which generate a MacSyFinder config file

```
class macsyfinder.scripts.macsyconfig.ConfigParserWithComments(defaults=None, dict_type=<class 'dict'>, allow_no_value=False, *,
    delimiters=('=', ':'),
    comment_prefixes=(';', '#'),
    inline_comment_prefixes=None,
    strict=True,
    empty_lines_in_values=True,
    default_section='DEFAULT',
    interpolation=<object object>,
    converters=<object object>,
    allow_unnamed_section=False)
```

Extend ConfigParser to allow comment in serialization

add_comment(*section: str, option: str, comment: str, comment_nb: int = count(1), add_space_before: bool = False, add_space_after: bool = True*) → None

Write a comment in .ini-format (start line with #)

Parameters

- **section** – the name of the section
- **option** – the name of the option
- **comment** – the comment linked to this option
- **comment_nb** – the identifier of the comment by default an integer
- **add_space_before**
- **add_space_after**

write(*file: IO*) → None

Write an .ini-format representation of the configuration state.

Parameters

file (*file*) – the file object wher to write the configuration

```
class macsyfinder.scripts.macsyconfig.Theme(ERROR: str = '\x1b[1m\x1b[31m', WARN: str =
\x1b[33m', SECTION: str = '\x1b[35m', RESET: str =
\x1b[0m', RETRY: str = '\x1b[33m', QUESTION: str =
\x1b[32m', EMPHASIZE: str = '\x1b[1m',
EXPLANATION: str = '\x1b[0m', DEFAULT: str =
\x1b[1m\x1b[32m')
```

Handle color combination to highlight interactive question

__delattr__(*name*)

Implement delattr(self, name).

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__init__(*ERROR: str = '\x1b[1m\x1b[31m', WARN: str = '\x1b[33m', SECTION: str = '\x1b[35m', RESET: str = '\x1b[0m', RETRY: str = '\x1b[33m', QUESTION: str = '\x1b[32m', EMPHASIZE: str = '\x1b[1m', EXPLANATION: str = '\x1b[0m', DEFAULT: str = '\x1b[1m\x1b[32m'*) → None

__repr__()

Return repr(self).

__setattr__(*name, value*)

Implement setattr(self, name, value).

__weakref__

list of weak references to the object

macsyfinder.scripts.macsyconfig._validator(*cast_func: Callable, raw: Any, default: Any, sequence: bool = False*) → Any

Parameters

- **cast_func** – the function which will cast the raw value
- **raw** – the raw value

- **default** – the default value
- **sequence** – True if the value is a sequence, False otherwise

Returns

The cast Value

Raises

MacsyppyError – if the raw value cannot be cast

```
macsyfinder.scripts.macsyconfig.ask(question: str, validator: Callable, default: Any = None, expected:  
Any = None, explanation: str = "", sequence: bool = False,  
question_color: str | None = None, retry: int = 2)
```

ask a question on the terminal and return the user response check if the user response is allowed (right type, among allowed values, ...)

Parameters

- **question** – The question to prompt to the user on the terminal
- **validator** – what validator to be used to check the user response
- **default** – the default value
- **expected** – the values allowed (can be a list of value)
- **explanation** – some explanation about the option
- **sequence** – True if the parameter accept a sequence of value (comma separated values)
- **question_color** – the color of the question display to the user
- **retry** – The number of time to repeat the question if the response is rejected

Returns

the value casted in right type

```
macsyfinder.scripts.macsyconfig.check_bool(raw: str, default: bool, expected, sequence: bool = False)  
→ bool
```

Check if value can be cast in str

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises

MacsyppyError – if the value cannot be cast in right type

```
macsyfinder.scripts.macsyconfig.check_choice(raw: str, default: str, expected: list[str], sequence: bool  
= False) → str
```

Check if value is in list of expected values

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – the allowed values for this option

- **sequence** – True if parameter accept a sequence of value, False otherwise

Returns

value

Raises**MacsyppyError** – if the value cannot be cast in right type

macsyfinder.scripts.macsyconfig.**check_dir**(raw: str, default: str, expected, sequence: bool = False) → str

Check if value point to a directory

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises**MacsyppyError** – if the value cannot be cast in right type

macsyfinder.scripts.macsyconfig.**check_exe**(raw: str, default: str, expected, sequence: bool = False) → str

Check if value point to an executable

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises**MacsyppyError** – if the value cannot be cast in right type

macsyfinder.scripts.macsyconfig.**check_file**(raw: str, default: str, expected, sequence: bool = False) → str

Check if value point to a file

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises**MacsyppyError** – if the value cannot be cast in right type

`macsyfinder.scripts.macsyconfig.check_float(raw: str, default: float, expected, sequence: bool = False)`
→ float

Check if value can be cast in float

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises

MacsyError – if the value cannot be cast in right type

`macsyfinder.scripts.macsyconfig.check_positive_int(raw: str, default: int, expected, sequence: bool = False)` → int

Check if value can be cast in integer >=0

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises

MacsyError – if the value cannot be cast in right type

`macsyfinder.scripts.macsyconfig.check_str(raw: str, default: str, expected, sequence: bool = False)` → str

Check if value can be cast in str

Parameters

- **raw** – the value return by the user
- **default** – the default value for the option
- **expected** – not used here to have the same signature for all check_xxx functions

Returns

value

Raises

MacsyError – if the value cannot be cast in right type

`macsyfinder.scripts.macsyconfig.epilog(path: str)` → str

Returns

the text to the user before to start the configuration

`macsyfinder.scripts.macsyconfig.main(args: list[str] | None = None)` → None

The main entryptoint of the script

Parameters

args – the command line arguments.

`macsyfinder.scripts.macsyconfig.parse_args(args: list[str]) → Namespace`
 parse command line

Parameters

args – the command line arguments

Returns

`macsyfinder.scripts.macsyconfig.prolog() → str`

Returns

the text displayed to the user when the configuration file is generated

`macsyfinder.scripts.macsyconfig.serialize(config: ConfigParserWithComments, path: str) → None`
 save the configuration on file

Parameters

- **config** – the config to save
- **path** (*str*) – where to store the configuration

`macsyfinder.scripts.macsyconfig.set_base_options(config: ConfigParserWithComments, defaults: MacsyDefaults, use_defaults: bool = False) → None`

Options for base section

Parameters

- **config** – The config to setup
- **defaults** – the macsyfinder defaults values
- **use_defaults** (*bool*) – If True do not ask any question use the defaults values

`macsyfinder.scripts.macsyconfig.set_general_options(config: ConfigParserWithComments, defaults: MacsyDefaults, use_defaults: bool = False) → None`

Options for general section

Parameters

- **config** – The config to setup
- **defaults** – the macsyfinder defaults values
- **use_defaults** (*bool*) – If True do not ask any question use the defaults values

`macsyfinder.scripts.macsyconfig.set_hmmer_options(config: ConfigParserWithComments, defaults: MacsyDefaults, use_defaults: bool = False) → None`

Options for hmmer section

Parameters

- **config** – The config to setup
- **defaults** – the macsyfinder defaults values
- **use_defaults** (*bool*) – If True do not ask any question use the defaults values

`macsyfinder.scripts.macsyconfig.set_path_options(config: ConfigParserWithComments, defaults: MacsyDefaults, use_defaults: bool = False) → None`

Options for directories section

Parameters

- **config** – The config to setup
- **defaults** – the macsyfinder defaults values
- **use_defaults** (*bool*) – If True do not ask any question use the defaults values

`macsyfinder.scripts.macsyconfig.set_score_options`(*config*: [ConfigParserWithComments](#), *defaults*: *MacsyDefaults*, *use_defaults*: *bool = False*) → *None*

Options for scoring section

Parameters

- **config** – The config to setup
- **defaults** – the macsyfinder defaults values
- **use_defaults** (*bool*) – If True do not ask any question use the defaults values

`macsyfinder.scripts.macsyconfig.set_section`(*sec_name*: *str*, *options*: *dict[slice(<class 'str'>, typing.Any, None)]*, *config*: *~macsyfinder.scripts.macsyconfig.ConfigParserWithComments*, *defaults*: *~macsylib.config.MacsyDefaults*, *use_defaults*: *bool = False*) → [ConfigParserWithComments](#)

iter over options of a section ask question for each option and set this option in the config

Parameters

- **sec_name** – the name of the section
- **options** – a dictionary with the options to set up for this section
- **config** – The config to fill in.
- **defaults** – the macsyfinder defaults values
- **use_defaults** – The user skip this section so use defaults to set in config object

Returns

configuration

macsyprofile

wrapper to `macsylib.macsyprofile`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

m

`macyfinder.scripts.macyconfig`, [106](#)
`macyfinder.scripts.macydata`, [106](#)
`macyfinder.scripts.macyprofile`, [112](#)
`macyfinder.scripts.msf`, [106](#)

Symbols

`__delattr__()` (*macsyfinder.scripts.macsyconfig.Theme* method), 107
`__eq__()` (*macsyfinder.scripts.macsyconfig.Theme* method), 107
`__hash__()` (*macsyfinder.scripts.macsyconfig.Theme* method), 107
`__init__()` (*macsyfinder.scripts.macsyconfig.Theme* method), 107
`__repr__()` (*macsyfinder.scripts.macsyconfig.Theme* method), 107
`__setattr__()` (*macsyfinder.scripts.macsyconfig.Theme* method), 107
`__weakref__` (*macsyfinder.scripts.macsyconfig.Theme* attribute), 107
`_validator()` (in module *macsyfinder.scripts.macsyconfig*), 107

A

`add_comment()` (*macsyfinder.scripts.macsyconfig.ConfigParserWithComments* method), 106
`alarm_handler()` (in module *macsyfinder.scripts.msf*), 106
`ask()` (in module *macsyfinder.scripts.macsyconfig*), 108

C

`check_bool()` (in module *macsyfinder.scripts.macsyconfig*), 108
`check_choice()` (in module *macsyfinder.scripts.macsyconfig*), 108
`check_dir()` (in module *macsyfinder.scripts.macsyconfig*), 109
`check_exe()` (in module *macsyfinder.scripts.macsyconfig*), 109
`check_file()` (in module *macsyfinder.scripts.macsyconfig*), 109
`check_float()` (in module *macsyfinder.scripts.macsyconfig*), 109
`check_positive_int()` (in module *macsyfinder.scripts.macsyconfig*), 110
`check_str()` (in module *macsyfinder.scripts.macsyconfig*), 110

`CITATION.yml`, 105

`Cluster`, 104

`ConfigParserWithComments` (class in *macsyfinder.scripts.macsyconfig*), 106

`CONTRIBUTING`, 105

`CONTRIBUTORS`, 105

`COPYING`, 105

`COPYRIGHT`, 105

D

`doc`, 104

E

`epilog()` (in module *macsyfinder.scripts.macsyconfig*), 110

L

`list_models()` (in module *macsyfinder.scripts.msf*), 106

M

`macsyfinder`, 104

`macsyfinder.scripts.macsyconfig` module, 106

`macsyfinder.scripts.macsydata` module, 106

`macsyfinder.scripts.macsyprofile` module, 112

`macsyfinder.scripts.msf` module, 106

`main()` (in module *macsyfinder.scripts.macsyconfig*), 110

`main()` (in module *macsyfinder.scripts.msf*), 106

`Model`, 104

`Model family`, 104

`ModelDefinition`, 104

`module`

`macsyfinder.scripts.macsyconfig`, 106

`macsyfinder.scripts.macsydata`, 106

`macsyfinder.scripts.macsyprofile`, 112

`macsyfinder.scripts.msf`, 106

O

`outfile_header()` (in module *macsyfinder.io*), 105

P

`parse_args()` (in module *macsyfinder.scripts.macsyconfig*), 110
`parse_args()` (in module *macsyfinder.scripts.msf*), 106
`prolog()` (in module *macsyfinder.scripts.macsyconfig*), 111
`pyproject.toml`, 105

R

`README.md`, 105

S

`serialize()` (in module *macsyfinder.scripts.macsyconfig*), 111
`set_base_options()` (in module *macsyfinder.scripts.macsyconfig*), 111
`set_general_options()` (in module *macsyfinder.scripts.macsyconfig*), 111
`set_hmmer_options()` (in module *macsyfinder.scripts.macsyconfig*), 111
`set_path_options()` (in module *macsyfinder.scripts.macsyconfig*), 111
`set_score_options()` (in module *macsyfinder.scripts.macsyconfig*), 112
`set_section()` (in module *macsyfinder.scripts.macsyconfig*), 112
`setup.py`, 105
`Solution`, 104
`System`, 104

T

`tests`, 104
`Theme` (class in *macsyfinder.scripts.macsyconfig*), 107

U

`utils`, 105

W

`write()` (*macsyfinder.scripts.macsyconfig.ConfigParserWithComments* method), 107