1. Pipeline Protocol 2: Model Comparison

1. Objective

To establish the standardized procedure for comparing Genome-Scale Metabolic Models (GEMs) for *in silico* analysis under a common simulation framework using the GEMcompare package.

2. Scope

Applies to all Genome-Scale Metabolic Models (GEMs) in Systems Biology Markup Language (SBML) and .XML format with BiGG identifiers and same culture medium.

3. Definitions

As previously defined in Protocol 1.

4. Responsibilities

- The student or professional is responsible for executing the procedure, documenting changes, and storing the models.
- The coordinator or director is responsible for supervising that the procedure is performed correctly.

5. Materials

5.1 Equipment

Standard personal computer with Python (version 3.11.11 tested) properly installed.

5.2 Software

- GEMcompare pipeline with ”Models” and ”Pipeline” folders
- Python packages: Cobra (version 0.29.1 or higher), mergem (version 1.1.0 or higher), libsbml (version 5.20.5 or higher), memote (version 0.17.0 or higher), PyYAML (version 6.0.1 or higher), pandas (version 2.3.1 or higher), jupyter notebook (version 7.4.7 or higher). All these packages can be installed directly in the terminal with the command: `pip install cobra>=0.29.1 mergem>=1.1.0 python-libsbml>=5.20.5 memote>=0.17.0 pyyaml>=6.0.1 pandas>=2.3.1 jupyter>=7.4.7`.
- Models to be compared in .XML format in Systems Biology Markup Language (SBML) in the ”GEMcompare/outputs/prepared_models” folder if they have already passed through the first part of the pipeline corresponding to preparation.
- Configuration file (config.yaml) included in the pipeline at ”GEMcompare/config.yaml”
- Excel file with binary dataset (essential/non-essential) of essentiality ”Pipeline/2_Comparison /Essentiality_dataset.xlsx”. This file is expected to have the following two columns

(exact names): ”genomic_annotation.locus_tag” and ”essential” as it was related in the dataset for *B. subtilis*.

- Excel file with defined culture media for the final media configuration step at ”Pipeline/2 /CarbonUtilization.xlsx”. This file must contain a sheet called ”Names” that relates carbon sources and their growth in binary (growth: +/no growth: -).
- Although not necessary, it is recommended to have an IDE or code editor such as Visual Studio Code installed.

6. Procedure

6.1 Bimodal Configuration: Command Line, Config File, or Default

The user is recommended to use option 6.1.2 with the config.yaml file as the main method to configure the pipeline as it is the simplest, most intuitive and easiest to modify. Remember to navigate to the directory folder where the pipeline is located by entering the command `cd .../GEMcompare` in the terminal. Once all previously mentioned materials are available, select only one of the following options.

6.1.1 CLI Configuration (Command-line Interface) To configure the pipeline via console and if you want to add parameters different from the default ones, arguments must be added for the following parameters:

- `--models`: path to models, one or more can be added separated by spaces.
- `--config` (optional): path to configuration file, by default it will always be config.yaml.
- `--skip` (optional): names of pipeline parts to skip. Options: structural, functional, diff_Carbonsources, GeneEssentiality, CarbonUtilization.
- `--verbose` (optional): used to show detailed process information.

A typical workflow to run the pipeline via console would be:

```
cd Downloads/GEMcompare
python Pipeline/2_Comparison.py --models Models/modelo1 Models/modelo2 -
-skip functional --verbose
```

6.1.2 Configuration File config.yaml The config.yaml file serves as the configuration file for the entire pipeline. In it we find sections with names alluding to the Comparison parts (structural, functional, carbon_sources, gene_essentiality, biolog). In it we find the ”models” section where we can list the models to be compared, as well as different attributes for each section such as:

- `Run`: Allows determining which pipeline parts to skip.
- `output_dir`: The directories where the results of each script will go.
- `output_name`: Names of the resulting files.
- `force`: if true it overwrites if previous files exist.

For the ”functional” section, the attribute `pathways` is found where a dictionary of reactions corresponding to the pathways of interest can be defined.

For the ”carbon_sources” and ”biolog” sections, the attribute `uptake` exists which represents the lower limit of the substrate consumption rate. The latter also has `threshold` which defines the limit to consider a result as growth.

The user is recommended to use this option to configure the pipeline.

6.1.3 Default Configuration Finally, the pipeline can be run as it is since the config.yaml file is already configured to be run by default. However, the models that will be compared this way are the three that come by default (iYO844, iBB1018 and iBsu1147R).

6.2 Stand-alone Modes

As in the previous Protocol 1, a part of the pipeline (a script) can be run individually to perform a specific analysis.

For all stand-alone modes the logic used will be the same, that is, you can use the script of the test to be performed by passing arguments via console. For example for any ”SCRIPT” a typical workflow would be:

```
python Pipeline/2_Comparison/2.X_SCRIPT.py --models Models/modelo1
Models/modelo2
--outdir GEMcompare/results --outname SCRIPT_modelos1y2 --verbose --force
--additional_parameters
```

The options for that ”SCRIPT” are:

- 2.1_Structural.py
- 2.2_Functional.py
- 2.3_Diff_CarbonSources.py
- 2.4_GeneEssentiality.py
- 2.5_CarbonUtilization.py

6.3 Running the Complete Pipeline

Once the pipeline is configured as indicated in 6.1, the user can run it in two ways. The first is from the console using the command:

```
python Pipeline/2_Comparison.py --models Models/modelo1 Models/modelo2
```

Or also:

```
python Pipeline/2_Comparison.py --config config.yaml
```

The other way is to locate yourself in your preferred IDE and clicking the ”Run” button where the config.yaml configuration file will be used by default.

6.4 Results

The result of this second part of the pipeline corresponding to model comparison will be the generation of graphs and metrics that allow outlining in a general way the differences between GEMs in a standardized manner, provided they have the expected formats and/or have passed through the preparation of the first protocol. This comparison allows the user to learn more about the GEMs and is a first step for building scripts that perform all types of more complex comparisons.