User Manual of AutoPACMEN

(**A**utomatic Integration of **P**rotein **A**llocation **C**onstraints in **Me**tabolic **N**etworks)

Table of Contents

1 Overview

AutoPACMEN (**A**utomatic Integration of **P**rotein **A**llocation **C**onstraints in **Me**tabolic **N**etworks) consists of two major parts (see Figure 1): The AutoPACMEN Model Generator takes standard constraint-based models (in SBML format) as input and generates sMOMENT models from them. The optional AutoPACMEN Model Calibrator supports calibration of the protein pool variable *P* and of the k_{cat} parameters. For a detailed description of the sMOMENT method and for further information o AutoPACMEN see the original work [Bekiaris PS, Klamt S (2019) "Automatic Construction of Metabolic Models with Enzyme Constraints", submitted].

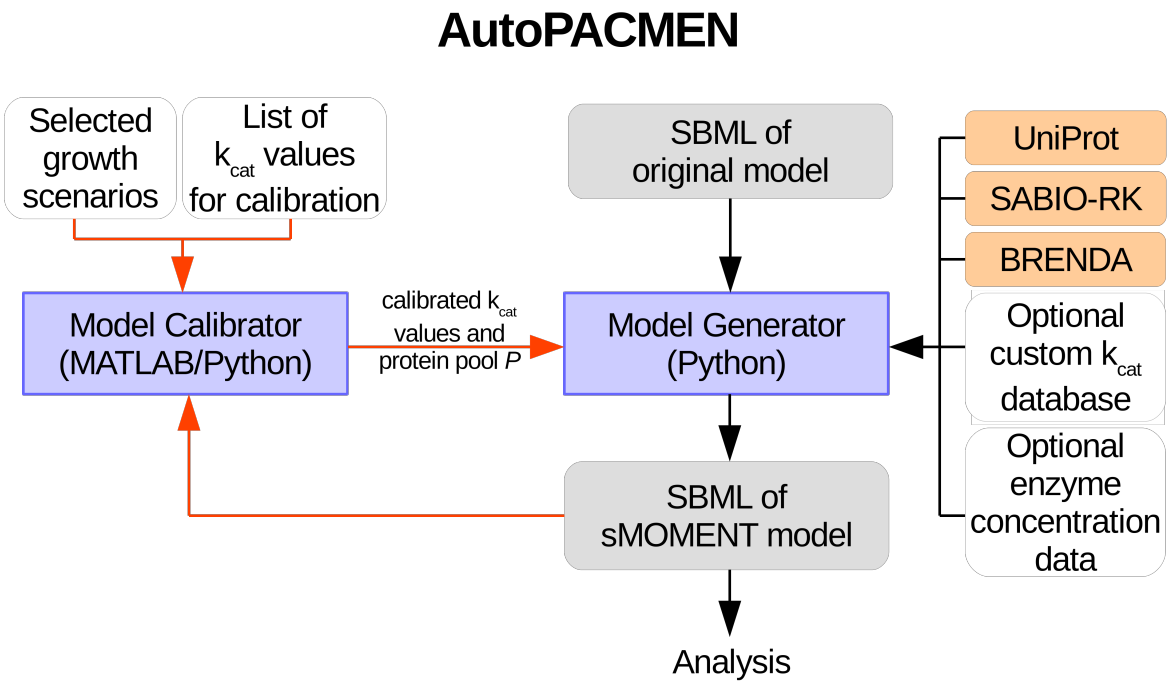

Figure 2 General overview of the structure and workflow of the AutoPACMEN toolbox consisting of the model generator and model calibrator. The red arrows show the optional model calibrator workflow.

All AutoPACMEN scripts are open source, licensed under the Apache license.

2 Requirements

The AutoPACMEN Model Generator is written in Python 3 and requires a Python version >=3.7. Aside of Python's standard library, the Model Generator also uses the modules biopython (1), cobrapy (2), click, openpyxl, pebble and xlsxwriter.

The AutoPACMEN Model Calibrator is partly written in Python 3 (with the same dependencies as the Model Generator) and partly written in MATLAB, requiring a MATLAB version >=2017a and using the MATLAB package *CellNetAnalyzer* (3).

3 Download Procedure

Recommended for most users of the Python modules:
You can install AutoPACMEN's Python modules with pip using:

```
pip install autopacmen-Paulocracy
```

Recommended for advanced users of the Python modules and all users of the MATLAB modules:
If you do not intend to install AutoPACMEN from pip and use AutoPACMEN's scripts directly, you can download AutoPACMEN from its GitHub repository at:
https://github.com/Paulocracy/autopacmen
Afterwards, continue with the installation procedure described in the next chapter.

4 Installation Procedure (without pip)

4.1 AutoPACMEN Model Generator and Model Calibrator Python scripts

Since the Model Generator and the Model Calibrator's Python scripts are solely in Python 3 only, it can theoretically run on every system which is supported by Python itself. This includes Microsoft Windows (c), Linux, BSD and MacOS. However, AutoPACMEN's Python scripts were only tested on Windows and Linux. In order to install Python, you can download it from its main website (https://www.python.org/, accessed on August 26, 2019) for free, and follow its installation instructions. It is useful to make Python callable from your system's console by adding it to its PATH variable.

The next step is the installation of the additional Python modules for this package. In order to install them, open your system's console and execute the following commands:

```
pip install biopython
pip install cobra
pip install click
pip install openpyxl
pip install pebble
pip install requests
pip install xlsxwriter
```

If everything went successfully, you can now use AutoPACMEN's Python scripts using Python 3.

4.2 AutoPACMEN Model Calibrator MATLAB scripts

The Model Calibrator's MATLAB scripts needs a working MATLAB installation on your system. MATLAB itself is a commercial and paid application and can be retrieved via its main website (https://www.mathworks.com/products/matlab.html, accessed on August 26, 2019). Additionally, the Model Calibrator needs the MATLAB package *CellNetAnalyzer* (3), which can be obtained from its own main website (http://www2.mpi-magdeburg.mpg.de/projects/cna/cna.html).

In order to use the Model Calibrator, one needs to start MATLAB with CellNetAnalyzer by starting MATLAB in CellNetAnalyzer's folder, using the command "startcna", and additionally adding MATLAB's path to the Model Calibrator's script files.

5 Usage of AutoPACMEN Model Generator

5.1 Introduction

All functions of the AutoPACMEN Model Generator can be used in 3 different ways:

1. As interactive console programs by starting a script in AutoPACMEN's main folder using the console without arguments. In this mode, the user is asked for every argument step by step and can enter them using the keyboard.
2. As console command by starting a program in AutoPACMEN's main folder with arguments.
3. As Python 3 modules, as done by all scripts in AutoPACMEN's main folder. The actual console-interface-independent modules can be found in be subfolder "submodules". The fully commented source code explains the usage of the single modules.

Whichever way is chosen, the basic workflow remains the same and is further explained in this chapter.

5.2 Retrieval of data from external data sources

The 1st step of the automatic generation of a model with protein allocation constraints is the retrieval of data from external databases. This includes the BIGG database for metabolites (4) as well as the enzyme databases BRENDA (5) and SABIO-RK (6).

These data scripts should be executed in the following order:

data_parse_bigg_metabolites_file.py

Usage: data_parse_bigg_metabolites_file.py [OPTIONS]

Converts the given BIGG metabolites text file into a machine-readable JSON file.

The BIGG metabolites text file can be downloaded as text file from http://bigg.ucsd.edu/data_access (accessed on August 22, 2019). This text file contains all metabolite ID data of the BIGG project. The newly created JSON will have the name 'bigg_id_name_mapping.json'.

Example:

Load the BIGG metabolites file 'C:\bigg_database.txt' and store the newly generated JSON in the folder 'C:\database\':

```
python data_parse_bigg_metabolites_file.py
--bigg_metabolites_file_path C:\bigg_database.txt
--json_output_folder C:\database
```

Options:

--bigg_metabolites_file_path TEXT

BIGG metabolites file path [required]

--json_output_folder TEXT Path to the folder in which the newly

generated JSON will be created [required]

--help Show this message and exit.

data_parse_brenda_textfile.py

Usage: data_parse_brenda_textfile.py [OPTIONS]

Converts the given BRENDA text file into a more machine-readable JSON file.

The BRENDA database can be downloaded as text file from https://www.brenda-enzymes.org/download_brenda_without_registration.php (accessed on August 6, 2019). This text file contains all data of BRENDA in an EC-number-sorted manner. Since the text file does not have an easily readable standardized format, this script converts it into a JSON which contains only the sMOMENT-relevant data in an ordered fashion. Additionally, substrate names of BRENDA are converted with BIGG metabolite names if the name of the BRENDA substrate is equal to a BIGG metabolite name.

Example:

Load the BRENDA textfile 'C:\folder\download.txt' while the BIGG metabolites file is in 'C:\bigg\' and the newly generated JSON shall be named 'C:\folder\brenda.json':

```
python data_parse_bigg_metabolites_file.py --brenda_textfile_path
C:\folder\download.txt --bigg_metabolites_json_folder C:\bigg\
--json_output_path C:\folder\brenda.json
```

Options:

--brenda_textfile_path TEXT Full path to the BRENDA database text file.

[required]

--bigg_metabolites_json_folder TEXT

The folder of the BIGG metabolites JSON

(created using

data_parse_bigg_metabolites_file.py). The

name of the JSON must not have been changed.

[required]

--json_output_path TEXT The full path for the more machine-readable

JSON file of the BRENDA database text file

shall be placed. The resulting JSON will

have the name. This JSON will be created

with this script. [required]

--help Show this message and exit.

data_parse_brenda_json_for_model.py

Usage: data_parse_brenda_json_for_model.py [OPTIONS]

Converts the given BRENDA JSON created with data_parse_brenda_textfile.py into a even more easily readable model-specific JSON.

This conversion is needed for all subsequent AutoPACMEN steps. The model-specific JSON contains all of the model's EC number entries contains a polished version of the BRENDA JSON without transferred EC numbers (i.e., a reassigned EC number) and other possible problems.

Example:

Create a model-specific JSON called 'C:\folder\brenda_model_specific.json' from the BRENDA JSON 'C:\folder\brenda.json' using the model 'C:\folder\model.xml':

```
python data_parse_bigg_metabolites_file.py --sbml_path
C:\folder\model.xml --brenda_json_path C:\folder\brenda.json
--json_output_path C:\folder\brenda_model_specific.json
```

Options:

--sbml_path TEXT Full path to the SBML with the model of which the BRENDA JSON will be derived. [required]

--brenda_json_path TEXT Full path to the BRENDA JSON created with data_parse_brenda_textfile.py [required]

--json_output_path TEXT The full path for the model-specific JSON file of the BRENDA JSON created with data_parse_brenda_textfile.py [required]

--help Show this message and exit.

data_parse_sabio_rk_for_model.py

Usage: data_parse_sabio_rk_for_model.py [OPTIONS]

Using the given model, its EC numbers are used for a search in the SABIO-RK database. The kcat results are stored as a JSON :D SABIO-RK is called using its API, so that a woking internet connection is needed in order to run this script. In addition to the metbolic model, a BIGG ID name JSON generated by data_parse_bigg_metabolites_file.py has to be given as input. The resulting JSON includes found kcats for the EC numbers, including EC numbers for which kcats were only found with wilcard searches and EC nubmers for which a direct result was found. Additionally, the resulting JSON includes the organism of the EC-number-associated kcats and the substrates of the kcat data, using BIGG ID names if possible.

Example:

Create a SABIO-RK JSON called 'C:\folder\sabio_rk.json' using the model 'C:\folder\model.xml' and the BIGG ID<->name mapping JSON 'C:\folder\bigg_ids.json':

```
python data_parse_sabio_rk_for_model.py –sbml_path
C:\folder\model.xml --json_output_path C:\folder\sabio_rk.json
--bigg_id_name_mapping_path C:\folder\bigg_ids.json
```

Options:

--sbml_path TEXT The path to the metabolic model SBML from which the EC numbers will be read out in order to search them in SABIO-RK. [required]

--json_output_path TEXT The path to the newly generated JSON that will include the kcat data from SABIO-RK. [required]

--bigg_id_name_mapping_path TEXT The path to the JSON [required]

--help Show this message and exit.

data_create_combined_kcat_database.py

Usage: data_create_combined_kcat_database.py [OPTIONS]

Combines the BRENDA and SABIO-RK JSONs into one big JSON which can be used by modeling_get_reactions_kcat_mapping.py

The BRENDA JSON is to have been created with data_parse_brenda_json_for_model.py, the SABIO-RK JSON with data_parse_sabio_rk_for_model.py. Only entries which did not result from wildcard searches will be used for the combined database.

Example:

Combine the BRENDA JSON 'C:\JSONS\brenda.json' and the SABIO-RK JSON 'C:\JSONS\sabio.json' into the JSON 'C:\JSONS\combined.json:

```
python data_create_combined_kcat_database.py
--sabio_rk_kcat_database_path C:\JSONS\brenda.json
--brenda_kcat_database_path C:\JSONS\sabio.json --output_path
C:\JSONS\sabio.json
```

Options:

--sabio_rk_kcat_database_path TEXT Full path SABIO-RK JSON created with data_parse_brenda_json_for_model.py

[required]

--brenda_kcat_database_path TEXT

[required]

--output_path TEXT Full path to the newly created combined JSON

[required]

--help Show this message and exit.

5.3 Retrieval and setting of enzymatic data for model

After the basic k_{cat} databases were set with the scripts described in the previous chapter, the actual generation of metabolic-network-dependent data can be performed.

modeling_get_initial_spreadsheets.py

Usage: modeling_get_initial_spreadsheets.py [OPTIONS]

Creates initial AutoPACMEN XLSX spreadsheets in which additional information can be entered by the use.

For the application of sMOMENT, the following spreadsheets are relevant (* stands for the given project name):

- *_enzyme_stoichiometries.xlsx: here, the user can enter intra-complex stoichiometries of the proteins which constitute the complex
- *_protein_data.xlsx: here, the user has to enter the data for the calculation of the protein pool. Optionally, the user can also enter proteomic concentration data for each protein.

Options:

--sbml_path TEXT Path to SBML. [required]

--project_folder TEXT Path to project. [required]

--project_name TEXT Name of project. [required]

--help Show this message and exit.

Example:

Suppose we want to create the initial spreadsheets for the SBML 'C:\\Test.xml' and have the project folder 'C:\\folder' and the project name 'project':

```
python modeling_get_initial_spreadsheets.py --sbml_path
C:\\Test.xml --project_folder C:\\folder --project_name project
```

modeling_get_protein_mass_mapping.py

Usage: modeling_get_protein_mass_mapping.py [OPTIONS]

Creates a JSON with the protein masses for all proteins given in the gene rules of the given metabolic model.

The protein masses are read out using the UniProt API. Therefore, the gene rules need protein names which can be found in UniProt. The newly created JSON will be stored in the given project folder and have the file name of the project plus '_protein_id_mass_mapping.json'.

Example:

Suppose we want to get the protein masses of the SBML 'C:\model.xml' and the project folder is 'C:\folder\' and the project name is 'exemplary', the command would be:

```
python modeling_get_protein_mass_mapping.py --sbml_path
C:\model.xml --project_folder C:\folder\ --project_name exemplary
```

Options:

--sbml_path TEXT Path to the SBML representation of the metabolic model of whose gene rules the protein masses shall be read out [required]

--project_folder TEXT Path to the folder in which the JSON with the protein mass data will be created [required]

--project_name TEXT Name of the project. This will be the prefix of the created JSON. [required]

--help Show this message and exit.

modeling_get_reactions_kcat_mapping.py

Usage: modeling_get_reactions_kcat_mapping.py [OPTIONS]

Generates the reaction<->kcat mapping for the given metabolic model.

The algorithm of the kcat selection process is explained in this program package's publication.

Details on the optional user-defined kcat database: It must be a JSON of the following format:

```
{
    "$PROTEIN_IDENTIFIER": {
        "kcats": [
            $FLOAT_LIST_OF_ASSOCIATED_KCAT_VALUES
        ],
        "direction": {
    "$ASSOCIATED_REACTION_OF_WHICH_THE_KCATS_WERE_TAKEN":
    "$DIRECTION_OF_REACTION (either forward or reverse)"
        }
    }
```

```
}
```

In other words, this JSON contains protein-dependent measured kcat values, and the reactions and their directions for which these kcat values were measured.

Example:

Suppose we want to get the reactions<->kcat mapping of the metabolic model described by the SBML 'C:\model\test.xml', the additional information from AutoPACMEN's data_*.py scripts is in the project folder 'C:\project\', the project name is 'example', the metabolic model represents *Escherichia coli*, the combined reaction<->kcat database is in 'C:\database\database.json' and an optional user-defined protein database is given and can be found under 'C:\database\user.json', and we want to select the k_{cat}s using the mean, then our command would be:

```
python modeling_get_reactions_kcat_mapping.py --sbml_path
C:\model\test.xml --project_folder C:\project\ --project_name
example --organism 'Escherichia coli' --kcat_database_path
C:\database\database.json --protein_kcat_database_path
C:\database\user.json --type_of_kcat_selection ‘mean’
```

Options:

--sbml_path TEXT Full path to the SBML of the metabolic model that shall be sMOMENT-enhanced [required]

--project_folder TEXT Path to the project folder in which the reactions<->kcat mapping JSON will be created. [required]

--project_name TEXT Name of the current project. The generated reactions<->kcat mapping JSON will have this name as prefix. [required]

--organism TEXT The scientific name of the organism that the metabolic model shall represent, e.g. 'Escherichia coli'.This name is used, together with NCBI TAXONOMY, for the taxonomy-dependent search of kcat values. [required]

--kcat_database_path TEXT Full path to the SABIO-RK&BRENDA kcat<->reaction JSON which was creates using data_create_combined_kcat_database.py

[required]

--protein_kcat_database_path TEXT

Full path to the custom user-defined kcat<->protein JSON. It must be a dictionary containing the protein names(as given in the metabolic network's gene rules) as keys and associated kcats as children. See this script's description for more.

[required]

--type_of_kcat_selection TEXT Can be "mean", "median" or "random". Refers to the selection of found kcats of a reaction. Is "mean" by default.

[required]

--help Show this message and exit.

5.4 Generation of enzyme-constraint-enhanced model

These script automatically apply the sMOMENT or the GECKO (7) method on a given stoichiometric model, using the enzymatic data that was retrieved with the previously executed scripts. All further described described scripts as well as the Model Calibrator only work with sMOMENT-enhanced models.

modeling_create_smoment_model.py

Usage: modeling_create_smoment_model.py [OPTIONS]

Applies the sMOMENT method on the given SBML. All AutoPACMEN scripts starting with "data_" have had to be run first in order to get all necessary data.

The sMOMENT method itself is described in its publication's method section and AutoPACMEN’s manual.

Example:

Suppose we want to apply sMOMENT on the metabolic model described by the SBML 'C:\models\model.xml', the sMOMENT-enhanced model SBML shall have the path 'model_new.xml' (it will be stored in the project folder), the project folder containing additional information is 'C:\project\', the project name is 'example', and we want to exclude the introduction of the protein pool pseudo-metabolite in the reactions 'CBD' and 'ACALD', and we want to choose the default kcat using the median, then our command would be:

```
python modeling_create_smoment_model.py –input_sbml_path
C:\models\model.xml --output_sbml_name model_new.xml –
```

```
project_folder C:\project\ --project_name example
--excluded_reactions CBD;ACALD --type_of_default_kcat_selection
'median'
```

Options:

```
  --input_sbml_path TEXT     Path to the SBML model which describes the
                             stoichiometric metabolic model that shall be
                             converted into a protein-constrained-enhanced
                             model  [required]
  --project_folder TEXT      Path to project folder. This folder needs to
                             include the reaction<->kcat mapping,the
                             protein<->mass mapping and the enzyme
                             stoichiometry spreadsheet (all file names have to
                             start with the project name).  [required]
  --project_name TEXT        Project name, which represents the start of the
                             project folder's model files  [required]
  --output_sbml_name TEXT    File name of the output SBML which describes the
                             protein-constraint-enhanced model. This output
                             SBML will be stored in the project folder.
                             [required]
  --excluded_reactions TEXT  Excluded reactions for which the pseudo-
                             metabolite of the protein pool shall not be
                             introduced. Must be semicolon-separated.
                             [optional]
  --type_of_default_kcat_selection TEXT
                               Can be "mean", "median" or "random". Refers
                               to the selection of found kcats of a
                               reaction. Is "mean" by default.  [required]
  --help                     Show this message and exit.
```

modeling_create_gecko_model.py

Usage: modeling_create_gecko_model.py [OPTIONS]

Applies the original GECKO method on the given SBML. All AutoPACMEN scripts starting with "data_" have had to be run first in order to get all necessary data.

The GECKO method itself is described in (7)

Example:

Suppose we want to apply GECKO on the metabolic model described by the SBML 'C:\models\model.xml', the GECKO-enhanced model SBML shall have the path 'model_new.xml' (it will be stored in the project folder), the project folder containing additional information is 'C:\project\', the project name is 'example', and we want to exclude the introduction of the protein pool pseudo-metabolite in the reactions 'CBD' and 'ACALD', then our command would be:

```
python modeling_create_gecko_model.py --input_sbml_path
C:\models\model.xml --output_sbml_name model_new.xml
--project_folder C:\project\ --project_name example
--excluded_reactions CBD;ACALD
```

Options:

```
  --input_sbml_path TEXT     Path to the SBML model which describes the
                             stoichiometric metabolic model that shall be
                             converted into a protein-constrained-enhanced
                             model  [required]
  --project_folder TEXT      Path to project folder. This folder needs to
                             include the reaction<->kcat mapping,the
                             protein<->mass mapping and the enzyme
                             stoichiometry spreadsheet (all file names have to
                             start with the project name).  [required]
  --project_name TEXT        Project name, which represents the start of the
                             project folder's model files  [required]
  --output_sbml_name TEXT    File name of the output SBML which describes the
                             protein-constraint-enhanced model. This output
                             SBML will be stored in the project folder.
                             [required]
  --excluded_reactions TEXT  Excluded reactions for which the pseudo-
                             metabolite of the protein pool shall not be
                             introduced. Must be semicolon-separated.
                             [optional]
  --help                     Show this message and exit.
```

5.5 Analysis and testing of sMOMENT-enhanced model

After a sMOMENT-enhanced model was created using the script described in the previous chapter, AutoPACMEN provides two scripts for a quick console-based FVA analysis. Of course, the sMOMENT-enhanced SBML models can be used with any metabolic modeling suite such as cobrapy (2) or *CellNetAnalyzer* (3).

analysis_fva_comparison.py

Usage: analysis_fva_comparison.py [OPTIONS]

FVA (Flux Variability Analysis) comparison with the given arguments, e.g. in order to check the validity of the geckoed model.

An FVA result summary is shown for the original SBML model as well as for the protein-allocation-constrained one. The output is generated by cobrapy.

Example:

Print a comparative FVA for the non-constrained model 'C:\original.xml' and the sMOMENT-model 'C:\pac.xml' and the objective 'ACALD':

```
python analysis_fva_comparison.py –sbml_original_path
C:\original.xml --sbml_protein_constrained_path C:\pac.xml
--objective ACALD
```

Options:

--sbml_original_path TEXT Full SBML path of original model without protein allocation constraints [required]

--sbml_protein_constrained_path TEXT Full SBML path of sMOMENT-enhanced model. [required]

--objective TEXT Objective of the comparaive FVA. [required]

--help Show this message and exit.

analysis_fva_prot_pool.py

Usage: analysis_fva_prot_pool.py [OPTIONS]

Runs an FVA (Flux Variability Analysis) and prints the result with the sMOMENT-enhanced model and the given protein pool bounds.

The FVA results are generated by cobrapy can give a hint how to fit the protein pool.

Example:

Run FVAs with the sMOMENT model 'C:\model.xml' with the protein bounds 0.5, 0.1 and 0.05 and the objective reaction ACALD:

```
python analysis_fva_prot_pool.py --sbml_path C:\model.xml
--protein_pool_bounds 0.5;0.1;0.05 --objective ACALD
```

Options:

--sbml_path TEXT Full path to the sMOMENT model SBML [required]

--protein_pool_bounds TEXT Protein pool bounds for which the FVAs are run, semicolon-separated. [required]

--objective TEXT Objective of the FVA.

--help Show this message and exit.

5.6 Manual optimization

optimization_apply_manual_changes.py

Usage: optimization_apply_manual_changes.py [OPTIONS]

Applies manually given kcat changes to the given sMOMENT model.

This application is given by multiplying the stoichiometry of the protein pool pseudo-metabolite in a reaction with the inverse of the given manual change factor. The *inverse* is used as it is intended to change the kcat and as the protein pool stoichiometry is MW/kcat (MW is the molecular weight).

Example:

Suppose we want to lower the kcat of the reverse reaction of ACALD by the factor 10, and to heighten the kcat of the irreversible reaction CBD by the factor 5, and the SBML of the model which shall be changed has the path 'C:\models\model.xml' and the newly created SBML with the manual changes has the path 'C:\models\model_new.xml', then the command would be:

```
python optimization_apply_manual_changes –sbml_path_input
C:\models\model.xml  --sbml_path_output C:\models\model_new.xml
--kcat_change_factors ACALD,reverse,1/10;CBD,,5
```

Options:

--sbml_path_input TEXT Full path to the SBML of which the kcats shall be changed manually [required]

--sbml_path_output TEXT Full path to the newly created SBML with the manual kcat changes [required]

--kcat_change_factors TEXT Textual description of the manual kcat changes. The format of this description is: . Basic format: basereaction_id,reaction_direction,chang e_factor;(...) Example: if one wants to lower

the kcat of the reverse reaction of ACALD by the factor 10, and to highen the kcat of the irreversible reaction CBD by the factor 5, then the textual description of the changes would be: ACALD,reverse,1/10;CBD,,5 [required]

--help Show this message and exit.

6 Usage of Model Calibrator

6.1 Python scripts

The AutoPACMEN's Model Calibrator's Python scripts allows one to select reactions whose deletion of their associated enzyme allocation constraint (i.e., the addition of the protein pool metabolite) leads to a change of a given flux value of a reaction. Furthermore, this reaction selection can be further constrained to reactions whose enzyme allocation constraint removal leads to a change in only one given scenario only (scenarios are further explained in chapter 3.6.3).

The Python scripts are "reaction_flux_control_by_scenario.py" and "get_differential_reactions.py" in AutoPACMEN's "submodules" folder. "reaction_flux_control_by_scenario.py" contains a fully commented Python function (see this script more information) for the determination of the effect of enzyme allocation constraint removals on each of the given scenarios.
"get_differential_reactions.py" then returns a list of all reactions whose constraint removal has an effect in one scenario only (this script is also fully commented, see this file for more information on its usage).

An exemplary run of the Python functions can be found in AutoPACMEN's main folder as script "ec_model_2019_06_25_CALIBRATOR_get_reaction_flux_control_and_differential_reactions.py" .

6.2 MATLAB scripts

The AutoPACMEN Model Calibrator's MATLAB scripts provide the capability to automatically optimize the total protein pool and the k_{cat} values and of a sMOMENT-enhanced model. It has to be used together with the MATLAB metabolic modeling package *CellNetAnalyzer (3)*. After *CellNetAnalyzer* is installed and started as described in its manual, you can use the model calibrator by adding the folder containing it to MATLAB's path.

Optimization of total protein pool

The model calibrator's main function for the optimization of the total protein pool *P* (in [g/gDW]) is:

```
[best_prot_pool, start_prot_pool] = smoment_prot_pool_optimization(cna_model,
scenarios, scenarios_matrix, lowest_value, highest_value)
```

As arguments, it takes a sMOMENT-enhanced metabolic model as *CellNetAnalyzer* model (cna_model), a list of scenario dictionaries (scenarios; explained later in this chapter), the scenario

matrix for coupled scenarios (scenarios_matrix; explained later in this chapter) and the lowest (lowest_value) and highest (highest_value) value that the optimized total protein pool shall have. smoment_prot_pool_optimization returns two numbers: the first one is best_prot_pool, which is the optimized protein pool, the second one (start_prot_pool) is the original protein pool value which can be useful for a comparison with best_prot_pool.

Optimization of k_{cat} values
The model calibrator's main function for the optimization of k_{cat} values is:

```
[best_kcats, start_kcats] = smoment_kcat_optimization(cna_model,
reactions_to_change, scenarios, scenarios_matrix, max_change_factor)
```

As arguments, it takes a sMOMENT-enhanced metabolic model as *CellNetAnalyzer* model (cna_model), a list of reaction names for which the k_{cat} shall be optimized (reactions_to_change), a list of scenario dictionaries (scenarios; explained later in this chapter), the scenario matrix for coupled scenarios (scenarios_matrix; explained later in this chapter) and the maximal change factor that is applied to the k_{cat} values (max_change_factor), i.e. how much higher or lower an optimized k_{cat} value can be compared to the original k_{cat} value. smoment_kcat_optimization returns two matrices: the first one is best_kcats, which is the list of optimized k_{cat} values in the order of the given optimized reactions, the second one (start_kcats) is the list of the original unoptimized k_{cat} value which can be useful for a comparison with best_kcats.

6.3 Scenarios and scenario matrix

The scenarios are given as dictionaries, i.e. in a JSON file, which can be loaded into MATLAB using jsondecode(filename), as follows (Values is capital letters starting with $ describe variable names, i.e., $SCENARIO_NAME could be a name such as 'glucose aerobe', or $REACTION_ID could be 'ACALD', 'PFK', ...):

```
{
    "$SCENARIO_NAME": {
        "setup": {
            "$REACTION_ID": {
                "lower_bound": $lower_bound,
                "upper_bound": $upper_bound
            }
        },
        "target": {
          "reaction": "$REACTION_ID",
          "value": $TARGET_VALUE,
        }
    }
```

```
}
```

In other words, the scenarios describe what target value of the target reaction shall be achieved with the model under the constraints set to the reactions in the setup. The target value is a numeric value that is intended to be achieved if a normal FBA is run with this model.

The scenario matrix can be given as follows:

```
{
     “SCENARIO_NAME_1”, “SCENARIO_NAME_2;
     “SCENARIO_NAME_3”, “NA;
}
```

The matrix is used to described coupled scenario results. In the given example, the target result of the scenario 1 is fixes when scenario 2 is run. In contrast, scenario 3 is run on its own, without making an other scenario dependent on its result.

An exemplary and descriptive usage of the model calibrator’s function for k_{cat} values can be found in chapter 4.

7 References

1. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009 Jun 1;25(11):1422–3.

2. Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Systems Biology. 2013 Aug 8;7(1):74.

3. Klamt S, Saez-Rodriguez J, Gilles ED. Structural and functional analysis of cellular networks with CellNetAnalyzer. BMC Systems Biology. 2007 Jan 8;1(1):2.

4. King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, et al. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res. 2016 Jan 4;44(D1):D515–22.

5. Jeske L, Placzek S, Schomburg I, Chang A, Schomburg D. BRENDA in 2019: a European ELIXIR core data resource. Nucleic Acids Res. 2019 Jan 8;47(D1):D542–9.

6. Wittig U, Kania R, Golebiewski M, Rey M, Shi L, Jong L, et al. SABIO-RK—database for biochemical reaction kinetics. Nucleic Acids Res. 2012 Jan 1;40(D1):D790–6.

7. Sánchez BJ, Zhang C, Nilsson A, Lahtvee P-J, Kerkhoven EJ, Nielsen J. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. Molecular Systems Biology. 2017 Aug 1;13(8):935.