metquest Documentation

Release 0.1.28

Aarthi Ravikrishnan

Jun 11, 2018

About MetQuest

1 Authors **3**

2 Acknowledgments **5**

2.1 Installation 5

2.2 metquest 7

Python Module Index **15**

MetQuest is a dynamic programming based algorithm for identifying all possible pathways from metabolic networks between the source and the target metabolites. MetQuest requires the genome-scale metabolic reconstructions, set of seed, source and target metabolites and the pathway length cut-off. MetQuest is compatible with Python 3 and is OS-independent.

CHAPTER 1

Authors

- Aarthi Ravikrishnan
- Meghana Nasre
- Karthik Raman

CHAPTER 2

Acknowledgments

This work was supported by the Indian Institute of Technology Madras grant ERP/1314/004/RESF/KARH to KR and the INSPIRE fellowship, Department of Science and Technology, Government of India to AR.

2.1 Installation

Use pip3 to install metquest from PyPI:

2.1.1 Via Python Package

```
pip3 install metquest
```

2.1.2 Direct installation

1. Install Python 3.4 or higher
2. Clone this repository to your computer using `git` or download the repository and decompress it.
3. Navigate to the folder where metquest is downloaded and type

```
python setup.py install
```

(Elevated `sudo` rights may be required depending on the platform. Replace python with python3 when multiple python distributions are found.)

2.1.3 Required python packages

1. cobra >= 0.11.3
2. numpy >= 1.14.3

3. scipy
4. python-libsbml
5. networkx >= 2.1

2.1.4 Input

Folder whose structure is as shown:

```
mainfolder/
|-Example1/
|   |-- SBML model(s) of metabolic networks # XML files of the metabolic␣
↪networks(COBRA-compatible)
|   |-- seed_mets.txt       # Text file containing the seed metabolites separated by␣
↪a newline
|   |-- source_mets.txt     # Text file containing the source metabolites separated␣
↪by a newline
|   |-- target_mets.txt     # Text file containing the target metabolites separated␣
↪by a newline
|   |-- cutoff.txt          # Text file containing the size cut-offs separated by a␣
↪newline
|-Example2/
|   ...
```

Kindly ensure that the SBML model has the field <model id> and the metabolites are prefixed with the model identifiers, for instance, if the model identifier is 'ecoli_core_model', and the seed metabolite is 'fum_c', the input text file should contain ecoli_core_model fum_c

2.1.5 Running MetQuest

From command line

MetQuest can be directly run from the terminal as

```
metquest.sh <path containing the input folder>
```

Navigate to the folder where metquest is installed and type

```
python execute_metquest.py <path containing the input folder>
```

(Replace python with python3 when multiple python distributions are found.)

From python console

```
>>> import metquest
>>> metquest.execute_all_codes()
```

When prompted, enter the path containing the folder with all the data files

Running examples

In the python console, type the following

```
>>> import metquest
>>> metquest.example.run_this_example()
```

This will run the example files.

2.2 metquest

2.2.1 metquest package

Subpackages

metquest.example package

Submodules

metquest.example.run_this_example module

run_this_example()
This function runs the example with E. coli iJO1366 model with the seed, source and target metabolite input provided.

Parameters **None** –

Returns

Return type None

Module contents

Submodules

metquest.construct_graph module

create_graph(*path_name_with_models*, *no_of_orgs*)
This function creates bipartite graph of the organisms based on the path provided and the number of organsisms. For instance, if a folder has 3 model files, and the number of organisms is 2, 3 (3C2) different bipartite graphs are created. The graph objects and the dictionary are saved as gpickle and pickle files respectively.

Parameters

- **path_name_with_models** (*str*) – Absolute path name of the folder containing the models.
- **no_of_orgs** (*int*) – Number of organisms to be used for creating the DiGraph.

Returns

- **H** (*NetworkX DiGraph Object*) – Bipartite graph consisting of internal and exchange reactions in organisms
- **full_name_map** (*dict*) – Dictionary mapping the adhoc reaction names to reaction names in the model

metquest.execute_metquest module

execute_all_codes()
This function executes all the codes including constructing graphs and executing metquest.

Parameters **None** –

Returns

Return type None

find_important_reactions(*all_reactions_involved*, *currenttarmet*, *seed_metabolites*, *namemap*, *G*)
This function determines the important reactions based on the pathways generated for the target metabolite.

Parameters

- **all_reactions_involved** (*list*) – list of all reactions found in all the pathways from source to target
- **currenttarmet** (*str*) – Current target metabolite
- **seed_metabolites** (*set*) – Set of seed metabolites including the source
- **namemap** (*dict*) – Dictionary mapping the adhoc reaction names to reaction names in the model
- **G** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network

Returns

Return type None

Notes

We define important reactions as those reactions which occur in almost all the pathways producing the target metabolite (apart from the reactions that are involved in the production of target metabolite and the uptake of seed metabolite)

find_jaccard_between_paths(*only_source_to_target*)
This function determines the jaccard values between the pathways generated from the source to the target.

Parameters **only_source_to_target** (*list*) – list of lists consisting of all pathways producing the target metabolite from the source

Returns

- **jaccard_values** (*list*) – list of all jaccard values (float) for all the pathway combinations
- **path_combinations** (*list*) – list of all pathway combinations corresponding to the jaccard values

Notes

Jaccard value J = (set(A).intersection(set(B)))/(set(A).union(set(B))) J = 1 indicates two sets are the same J = 0 indicates two sets are different

find_pathways_involving_exchange_mets(*number_of_xml*, *pathway_table*, *currenttarmet*, *seed_metabolites*, *namemap*, *G*)
This function identifies the pathways producing the target metabolites, which involve exchange metabolites. This function prints output only when a community of organisms is considered, i.e., when more than one metabolic network is used.

Parameters

- **number_of_xml** (*int*) – Number of xml files in the folder
- **pathway_table** (*dict*) – Dictionary of dictionary containing the pathways of different sizes identified for every metabolite. This will have only the acyclic/ branched pathways.
- **currenttarmet** (*str*) – Current target metabolite
- **seed_metabolites** (*set*) – Set of seed metabolites including the source
- **namemap** (*dict*) – Dictionary mapping the adhoc reaction names to reaction names in the model
- **G** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network

Returns **exchange_candidates_inverted_dict** – Dictionary containing the number of times an exchange reaction is repeated

Return type dict

find_pathways_starting_from_source(*source_metabolites*, *pathway_table*, *currenttarmet*, *cutoff*, *G*)

This function finds all pathways starting from the source metabolites

Parameters

- **source_metabolites** (*list*) – List of source metabolites
- **pathway_table** (*dict*) – Dictionary of dictionary containing the pathways of different sizes identified for every metabolite. This will have only the acyclic/ branched pathways.
- **currenttarmet** (*str*) – Current target metabolite
- **cutoff** (*int*) – Maximum pathway length cutoff
- **G** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network

Returns

- **most_different_paths** (*dict*) – For the given source metabolite, a combination of two most different pathways based on minimum Jaccard value is returned.
- **only_source_to_target** (*list*) – list of list containing all pathways starting from source metabolite

print_summary(*scope*, *currenttarmet*, *pathway_table*, *cutoff*, *cyclic_pathways*, *namemap*, *source_metabolites*, *seed_metabolites*, *number_of_xml*, *G*)

This function prints the results summary obtained from the pathways, i.e., 1. Number of metabolites in scope 2. Target metabolite 3. Pathway size cutoff 4. Number of all branched pathways found from seed 5. Number of all branched pathways from seed whose size <= Pathway size cutoff 6. Minimum number of steps to produce target metabolite 7. Number of branched pathways from source whose size <= Pathway size cutoff 8. Target metabolite can be produced using cyclic pathway 9. Number of cyclic pathways whose size <= Pathway size cutoff 10. One of the combination of most different pathways producing target metabolite 11. Important reactions based on the frequency of occurrences

Parameters

- **scope** (*set*) – Set of metabolites that can be produced from the given set of seed metabolites
- **currenttarmet** (*str*) – Current target metabolite
- **pathway_table** (*dict*) – Dictionary of dictionary containing the pathways of different sizes identified for every metabolite. This will have only the acyclic/ branched pathways.

- **cutoff** (*int*) – Maximum pathway length cutoff
- **cyclic_pathways** (*dict*) – Dictionary of dictionary containing cyclic pathways of different sizes identified for every metabolite.
- **namemap** (*dict*) – Dictionary mapping the adhoc reaction names to reaction names in the model
- **source_metabolites** (*list*) – List of source metabolites
- **seed_metabolites** (*set*) – Set of seed metabolites including the source
- **number_of_xml** (*int*) – Number of xml files in the folder
- **G** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network

Returns

Return type None

write_output_to_file(*pathway_table*, *currenttarmet*, *cutoff*, *cyclic_pathways*, *folder_to_create*, *namemap*, *source_metabolites*, *G*)

This function writes the pathways of sizes less than or equal to the cutoff from source to the target and seed metabolites to target. This function also writes cyclic pathways of sizes less than or equal to cutoff from the source to target.

Parameters

- **pathway_table** (*dict*) – Dictionary of dictionary containing the pathways of different sizes identified for every metabolite. This will have only the acyclic/ branched pathways.
- **currenttarmet** (*str*) – Current target metabolite
- **cutoff** (*int*) – Maximum pathway length cutoff
- **cyclic_pathways** (*dict*) – Dictionary of dictionary containing cyclic pathways of different sizes identified for every metabolite.
- **folder_to_create** (*str*) – Name of the folder where results have to be written
- **namemap** (*dict*) – Dictionary mapping the adhoc reaction names to reaction names in the model
- **source_metabolites** (*list*) – List of source metabolites
- **G** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network

Returns

Return type None

metquest.fetch_reactions module

segregate_reactions_from_models(*path_name*)

This function gets the data pertaining to the reactions and the metabolites from the models of multiple organisms. This requires as input the pathname where the '.xml' files are located. From this path, this function reads all the files using the functions in the COBRA toolbox and generates the stoichiometric model for these SBML models.

Parameters **path_name** (*str*) – full path name where the model files are

Returns

- **all_organisms_info** (*dict*) – Dictionary of all model data (reaction information about all the organisms)

- **namemap** (*dict*) – Dictionary mapping the adhoc reaction names to reaction names in the model

metquest.generate_partitions module

generate_partitions (*maximumvalue*, *lbnumlist*, *columnvalue*)

This code takes as input the columnvalue (j), values of the shortest path of each of the metabolites (given as a list) and the sum that has to be obtained using these combination of numbers.

Parameters

- **maximumvalue** (*int*) – Maximum values which the numbers can take
- **lbnumlist** (*list*) – a list of values pertaining to the length of shortest paths of every metabolite
- **columnvalue** (*int*) – Desired sum to be obtained All the partitions of numbers which will generate the desired sum whose values are between the values for shortest paths and the maximum values.

Returns **all_partitions**

Return type List of tuples

Notes

For instance, if the column value is 7, the number of imputs is 2, and the shortest path of the metabolites is 4,3 respectively, and the maximum sum that has to be obtained is 8, then

```
>>> generate_partitions(7,[4,3],8)
[(4, 4), (5, 3)]
```

```
>>> generate_partitions(4, [2,1,1], 5)
[(2, 1, 2), (2, 2, 1), (3, 1, 1)]
```

metquest.get_reaction_types module

find_different_reaction_types (*stoi_matrix*, *model*, *current_model_name*)

This function finds the exchange, irreversible and the reversible reactions from the model.

Parameters

- **stoi_matrix** (*numpy array*) – full path name where the model files are
- **model** (*COBRA model object*) – COBRA model object created from SBML models
- **current_model_name** (*str*) – Name which is to be prefixed against every reaction/metabolite (to differentiate the entries in multiple organisms, when a community model is built)

Returns

- **exchange_met_ids** (*list*) – Metabolite identifiers of exchange metabolites
- **irrev_lhs_nodes** (*list*) – Metabolite identifiers of reactants of irreversible reactions
- **irrev_rhs_nodes** (*list*) – Metabolite identifiers of products of irreversible reactions
- **rev_lhs_nodes** (*list*) – Metabolite identifiers of reactants of reversible reactions

- **rev_rhs_nodes** (*list*) – Metabolite identifiers of products of reversible reactions
- **exchange_rxn_ids** (*list*) – Reaction identifers of exchange reactions
- **irrev_rxn_ids** (*list*) – Reaction identifiers of irreversible reactions
- **rev_rxn_ids** (*list*) – Reaction identifiers of reversible reactions

metquest.guided_bfs module

forward_pass(*graph_object*, *seedmets*)

This function carries out the Guided Breadth First Search on a directed bipartite graph starting from the entries in seed metabolite set.

Parameters

- **graph_object** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network
- **seedmets** (*set*) – Set of seed metabolites including the source

Returns

- **lower_bound_metabolite** (*defaultdict*) – Minimum number of steps required to reach a metabolite
- **status_dict** (*defaultdict*) – Dictionary pertaining to the status of every reaction - whether it has been visited or not
- **scope** (*set*) – Set of metabolites that can be produced from the given set of seed metabolites

Notes

Starting with the set of seed metabolites S, the algorithm first finds all the reactions from the set R, whose precursor metabolites are in S. Such reactions are marked visited and added to the visited reaction set. Metabolites produced by these reactions are checked. The reactions where these metabolites participate are then checked for the presence of all its predecessors and are added to the queue. This traversal continues in a breadth-first manner and stops when there are no further reactions to be visited.

metquest.package_data module

metquest.pathway_assembler module

find_pathways(*G*, *seed_mets_input*, *path_len_cutoff*, **args*)

This function tries to identify pathways between a set of seed and target metabolites of a given size cut-off.

Parameters

- **G** (*NetworkX DiGraph Object*) – Bipartite graph of the metabolic network
- **seed_mets_input** (*set*) – Set of seed metabolites including the source
- **path_len_cutoff** (*int*) – Maximum size of the pathways
- ***args** – Used to decide if a particular combination has to be evaluated or not. i.e., if the number of pathways produced for two different metabolites are higher, for instance, if in the reaction A + B -> C, A has 2000 pathways and B has 5 pathways, then C will have 10000 pathways at the maximum. If this pathway cutoff (maxnumpath) is 1000, this combination will not be evaluated, provided C has been already found. By default, it is set to 1000

Returns

- **pathway_table** (*dict*) – Dictionary of dictionary containing the pathways of different sizes identified for every metabolite. This will have only the acyclic/ branched pathways.
- **cyclic_pathways** (*dict*) – Dictionary of dictionary containing cyclic pathways of different sizes identified for every metabolite.
- **scope** (*set*) – Set of metabolites which can be synthesised

Module contents

Python Module Index

m

metquest, 13
metquest.construct_graph, 7
metquest.example, 7
metquest.example.run_this_example, 7
metquest.execute_metquest, 8
metquest.fetch_reactions, 10
metquest.generate_partitions, 11
metquest.get_reaction_types, 11
metquest.guided_bfs, 12
metquest.package_data, 12
metquest.pathway_assembler, 12

Index

C

create_graph() (in module metquest.construct_graph), 7

E

execute_all_codes() (in module metquest.execute_metquest), 8

F

find_different_reaction_types() (in module metquest.get_reaction_types), 11
find_important_reactions() (in module metquest.execute_metquest), 8
find_jaccard_between_paths() (in module metquest.execute_metquest), 8
find_pathways() (in module metquest.pathway_assembler), 12
find_pathways_involving_exchange_mets() (in module metquest.execute_metquest), 8
find_pathways_starting_from_source() (in module metquest.execute_metquest), 9
forward_pass() (in module metquest.guided_bfs), 12

G

generate_partitions() (in module metquest.generate_partitions), 11

M

metquest (module), 13
metquest.construct_graph (module), 7
metquest.example (module), 7
metquest.example.run_this_example (module), 7
metquest.execute_metquest (module), 8
metquest.fetch_reactions (module), 10
metquest.generate_partitions (module), 11
metquest.get_reaction_types (module), 11
metquest.guided_bfs (module), 12
metquest.package_data (module), 12
metquest.pathway_assembler (module), 12

P

print_summary() (in module metquest.execute_metquest), 9

R

run_this_example() (in module metquest.example.run_this_example), 7

S

segregate_reactions_from_models() (in module metquest.fetch_reactions), 10

W

write_output_to_file() (in module metquest.execute_metquest), 10