padmet-utils Documentation

Meziane AITE

Jan 24, 2020

Contents

CHAPTER 1

Installation

Clone repository from github:

```
git clone https://github.com/AuReMe/padmet-utils.git
```

CHAPTER 2

Usage

To get started using padmet, install the library as described above. Once the library becomes available on the given system, it can be developed against. The developed scripts do not need to reside in any particular location on the system.

CHAPTER 3

Development

Anyone interested in contributing or tweaking the library is more then welcome to do so. To start, simply fork the Git repository on Github and start playing with it. Then, issue pull requests.

CHAPTER 4

API Documentation

4.1 scripts API

4.1.1 Scripts: Connection

Description:

#TODO

biggAPI_to_padmet

Description: Require internet access !

Allows to extract the bigg database from the API to create a padmet.

1./ Get all reactions universal id from http://bigg.ucsd.edu/api/v2/universal/reactions, escape reactions of biomass.

2./ Using async_list, extract all the informations for each reactions (compounds, stochio, name . . .)

3./ Need to use sleep time to avoid to lose the server access.

4./ Because the direction fo the reaction is not set by default in bigg. We get all the models where the reaction is and the final direction will the one found in more than 75%

5./ Also extract xrefs

```
usage:
    biggAPI_to_padmet.py --output=FILE [--pwy_file=FILE] [-v]

options:
    -h --help     Show help.
    --output=FILE    path to output, the padmet file.
```

(continues on next page)

(continued from previous page)

```
    --pwy_file=FILE   add kegg pathways from pathways file, line:'pwy_id, pwy_name, x,
↪ rxn_id'.
    -v   print info.
```

padmet_utils.connection.biggAPI_to_padmet.**main**()

check_orthology_input

Description: Before running orthology based reconstruction it is necessary to check if the metabolic network and the proteom of the model organism use the same ids for genes (or at least more than a given cutoff). To only check this. Use the 2nd usage.

If the genes ids are not the same, it is necessary to use a dictionnary of genes ids associating the genes ids from the proteom to the genes ids from the metabolic network.

To create the correct proteom from the dictionnnary, use the 3nd usage Finnaly by using the 1st usage, it is possible to:

1/ Check model_faa and model_metabolic for a given cutoff

2/ if under the cutoff, convert model_faa to the correct one with dict_ids_file

3/ if still under, SystemExit()

```
usage:
    check_orthology_input.py    --model_metabolic=FILE    --model_faa=FILE    [--
↪cutoff=FLOAT] [--dict_ids_file=FILE] --output=FILE    [-v]
    check_orthology_input.py    --model_metabolic=FILE    --model_faa=FILE    [--
↪cutoff=FLOAT] [-v]
    check_orthology_input.py    --model_faa=FILE    --dict_ids_file=FILE    --
↪output=FILE [-v]

option:
    -h --help    Show help.
    --model_metabolic=FILE    pathname to the metabolic network of the model (sbml).
    --model_faa=FILE    pathname to the proteom of the model (faa)
    --cutoff=FLOAT    cutoff [0:1] for comparing model_metabolic and model_faa.␣
↪[default: 0.70].
    --dict_ids_file=FILE    pathname to the dict associating genes ids from the model_
↪metabolic to the model_faa. line = gene_id_in_metabolic_network       gene_id_in_faa
    --output=FILE    output of get_valid_faa (a faa) or get_dict_ids (a dictionnary␣
↪of gene ids in tsv)
    -v   print info
```

padmet_utils.connection.check_orthology_input.**main**()

enhanced_meneco_output

Description: The standard output of meneco return ids of reactions corresponding to the solution for gapfilling.

The ids are those from the sbml and so they are encoded.

This script extract the solution corresponding to the union of reactions “Computing union of reactions from all completion” Based on padmetRef return a file with more information for each reaction.

ex: RXN__45__5

RXN-5, common_name, ec-number, Formula (with id),Formula (with cname),Action,Comment Also, the output can be used as input of the script update_padmetSpec.py In the column Action: 'add' => To add the reaction, '' => to do nothing

Comment: the reason of adding the reaction (ex: added for gap-filling by meneco)

```
usage:
    enhanced_meneco_output.py --meneco_output=FILE --padmetRef=FILE --output=FILE [-v]

options:
    -h --help     Show help.
    --meneco_output=FILE    pathname of a meneco run' result
    --padmetRef=FILE    path to padmet file corresponding to the database of␣
↪reference (the repair network)
    --output=FILE    path to tsv output file
```

`padmet_utils.connection.enhanced_meneco_output.main()`

extract_orthofinder

Description: After running orthofinder on n fasta file, read the output file 'Orthogroups.csv'

Require a folder 'orthology_based_folder' with this archi:

|– **model_a** – model_a.sbml

|– **model_b** –model_b.sbml

And the name of the studied organism 'study_id'

1. Read the orthogroups file, extract orthogroups in dict 'all_orthogroups', and all org names
2. In orthology folder search for sbml files 'extension = .sbml'
3. For each models regroup all information in a dict dict_data:

   {'study_id': study_id, 'model_id' : model_id, 'sbml_template': path to sbml of model', 'output': path to the output sbml, 'verbose': bool, if true print information }

   The output is by default: output_orthofinder_from_'model_id'.sbml
4. Store all previous dict_data in a list all_dict_data
5. iter on dict from all_dict_data and use function dict_data_to_sbml

Use a dict of data dict_data and dict of orthogroups dict_orthogroup to create sbml files.

dict_data and dict_orthogroup are obtained with fun orthofinder_to_sbml

6./ Read dict_orthogroups and check if model associated to dict_data and study org share orthologue

7./ Read sbml of model, parse all reactions and get genes associated to reaction.

8./ For each reactions:

Parse genes associated to sub part (ex: (gene-a and gene-b) or gene-c) = [(gene-a,gene-b), gene-c]

Check if study org have orthologue with at least one sub part (gene-a, gene-b) or gene-c

if yes: add the reaction to the new sbml and change genes ids by study org genes ids

Create the new sbml file.

```
usage:
    extract_orthofinder --sbml=FILE/DIR --orthologues=DIR --study_id=STR --output=DIR␣
↪[--workflow=STR] [-v]
    extract_orthofinder --sbml=DIR --orthogroups=FILE --study_id=STR --output=DIR [--
↪workflow=STR] [-v]

option:
    -h --help    Show help.
    --sbml=DIR   Folder with sub folder named as models name within sbml file name as␣
↪model_name.sbml
    --orthogroups=FILE   Output file of Orthofinder run Orthogroups.tsv
    --orthologues=DIR   Output directory of Orthofinder run Orthologues
    --study_id=ID   name of the studied organism
    --workflow=ID   worklow id in ['aureme','aucome']. specific run architecture␣
↪where to search sbml files
   --output=DIR   folder where to create all sbml output files
    -v   print info
```

padmet_utils.connection.extract_orthofinder.**main**()

extract_rxn_with_gene_assoc

Description: From a given sbml file, create a sbml with only the reactions associated to a gene.

Need for a reaction, in section 'note', 'GENE_ASSOCIATION': . . . .

```
usage:
    extract_rxn_with_gene_assoc.py --sbml=FILE --output=FILE [-v]

options:
    -h --help     Show help.
    --sbml=FILE    path to the sbml file
    --output=FILE    path to the sbml output (with only rxn with genes assoc)
    -v   print info
```

padmet_utils.connection.extract_rxn_with_gene_assoc.**main**()

gbk_to_faa

Description: convert GBK to FAA with Bio package

```
usage:
    gbk_to_faa.py    --gbk=FILE --output=FILE [--qualifier=STR] [-v]

option:
    -h --help    Show help.
    --gbk=FILE    path to the gbk file.
    --output=FILE    path to the output, a FAA file.
    --qualifier=STR    the qualifier of the gene id [default: locus_tag].
    -v   print info
```

padmet_utils.connection.gbk_to_faa.**main**()

gene_to_targets

Description: From a list of genes, get from the linked reactions the list of products.

R1 is linked to G1, R1 produces M1 and M2. output: M1,M2. Takes into account reversibility

```
usage:
    gene_to_targets.py --padmetSpec=FILE --genes=FILE --output=FILE [-v]

option:
    -h --help     Show help
    --padmetSpec=FILE    path to the padmet file
    --genes=FILE   path to the file containing gene ids, one id by line
    --output=FILE    path to the output file containing all tagerts which can by␣
↪produced by all reactions associated to the given genes
    -v   print info
```

`padmet_utils.connection.gene_to_targets.main()`

modelSeed_to_padmet

Description: #TODO

```
usage:
    modelSeed_to_padmet.py --output=FILE --rxn_file=FILE --pwy_file=FILE [-v]

options:
    -h --help     Show help.
    --output=FILE    path of the padmet file to create
    --rxn_file=FILE   path to json file of modelSeed reactions
    --pwy_file=FILE   path to pathway reactions association from modelSeed
    -v   print info.
```

`padmet_utils.connection.modelSeed_to_padmet.main()`

padmet_to_asp

Description: Convert PADMet to ASP following these predicats: common_name({reaction_id or enzyme_id or pathway_id or compound_id} , common_name) direction(reaction_id, reaction_direction). reaction_direction in[LEFT-TO-RIGHT,REVERSIBLE] ec_number(reaction_id, ec(x,x,x)). catalysed_by(reaction_id, enzyme_id). uniprotID(enzyme_id, uniprot_id). #if has has_xref and db = "UNIPROT" in_pathway(reaction_id, pathway_id). reactant(reaction_id, compound_id, stoechio_value). product(reaction_id, compound_id, stoechio_value). is_a(compound_id, class_id). is_a(pathway_id, pathway_id).

```
usage:
    padmet_to_asp.py --padmet=FILE --output=FILE [-v]

option:
    -h --help     Show help.
    --padmet=FILE    path to padmet file to convert.
    --output=FILE    path to output file in lp format.
    -v    print info.
```

`padmet_utils.connection.padmet_to_asp.main()`

padmet_to_matrix

Description: Create a stoichiometry matrix from a padmet file.

The columns represent the reactions and rows represent metabolites.

S[i,j] contains the quantity of metabolite 'i' produced (negative for consumed) by reaction 'j'.

```
usage:
    padmet_to_matrix.py --padmet=FILE --output=FILE

option:
    -h --help    Show help.
    --padmet=FILE    path to the padmet file to convert.
    --output=FILE    path to the output file, col: rxn, row: metabo, sep = "         ".
```

`padmet_utils.connection.padmet_to_matrix.`**`main`**`()`

padmet_to_padmet

Description: Allows to merge 1-n padmet. 1./ Update the 'init_padmet' with the 'to_add' padmet(s). to_add can be a file or a folder with only padmet files to add.

padmetRef can be use to ensure data uniformization.

```
usage:
    padmet_to_padmet.py --to_add=FILE/DIR --output=FILE [--padmetRef=FILE]  [-v]

options:
    -h --help     Show help.
    --to_add=FILE/DIR    path to the padmet file to add (sep: ;) or path to folder of␣
↪padmet files.
    --output=FILE    path to the new padmet file
    --padmetRef=FILE    path to the padmet file representing to the database of␣
↪reference (ex: metacyc_18.5.padmet)
    -v   print info
```

`padmet_utils.connection.padmet_to_padmet.`**`main`**`()`

padmet_to_tsv

Description: convert a padmet representing a database (padmetRef) and/or a padmet representing a model (padmet-Spec) to tsv files for askomics.

1./ Folder creation given the output directory. Create this directory if required and create a folder padmetRef filename and/or padmetSpec filename

2./

2.1/ For padmetRef:

2.1.a/ Nodes get all reactions nodes => extract data from misc with extract_nodes(rxn_nodes, "reaction", "../rxn.tsv")

get all compounds nodes => extract data from misc with extract_nodes(cpd_nodes, "compounds", "../cpd.tsv")

get all pathways nodes => extract data from misc with extract_nodes(pwy_nodes, "pathway", "../pwy.tsv") get all xrefs nodes => extract data from misc with extract_nodes(xref_nodes, “xref”, “../xref.tsv”)

2.1.b/ Relations for each rxn in rxn nodes:

get all rlt consumes/produces => create list of data with extract_rxn_cpd(rxn_cpd_rlt)
fieldnames = “rxn_cpd”,”concerns@reaction”,”consumes@compound”,”produces@compound”,”stoichiometry”,

get all rlt is_in_pathway => create list of data with extract_rxn_pwy(rxn_pwy_rlt)
fieldnames = “rxn_pwy”,”concerns@reaction”,”in_pwy@pathway”

get all rlt has_xref => create list of data with extract_entity_xref(rxn_xref_rlt)

for each cpd in cpd nodes:

get all rlt has_xref => update previous list of data with extract_entity_xref(cpd_xref_rlt)
fieldnames = “entity_xref”,”concerns@reaction”,”concerns@compound”,”has_xref@xref”

```
usage:
    padmet_to_tsv.py --padmetSpec=FILE [--padmetRef=FILE] --output_dir=DIR [-v]
    padmet_to_tsv.py --padmetRef=FILE [--padmetSpec=FILE] --output_dir=DIR [-v]

options:
    -h --help     Show help.
    --padmetSpec=FILE    path of the padmet representing the network to convert
    --padmetRef=FILE    path of the padmet representing the database
    --output_dir=DIR
    -v
```

`padmet_utils.connection.padmet_to_tsv.main()`

pgdb_to_padmet

Description:

Read a PGDB folder (from BIOCYC/PATHWAYTOOLS) and create a padmet. 1./ To create a padmet without any genes information extracted use the first usage with:

pgdb: path to pgdb folder output: path to the padmet to create version: to specify the version of the pgdb (20.0, 22.0) db: to sepcify the name of the database (METACYC, ECOCYC, . . .) enhance: to also read the file metabolic-reaction.xml and add the to the padmet

2./ To create a padmet and add only reactions from pgdb if they are in padmetRef specifie. Copy information of the reaction not from the pgdb but from the padmetRef. This allow to uniform reaction to the same version of metacyc represented in the padmetRef For example, in some case 2 pgdb from different version can contain different information for a same reaction,pathway. . . In this case use:

padmetRef: path to the padmet of reference

3./ To create a padmet wth genes information extracted use: extract-gene

3.1/ To remove from the final padmet all reactions without genes associated use: no-orphan

4./ To read the metabolic-reaction.xml file, a sbml with some missing reactions in PGDB use: enhance

For more information of the parsing process read information below.

classes.dat: For each class: create new node / class = class UNIQUE-ID (1) => node.id = UNIQUE-ID COMMON-NAME (0-n) => node.Misc[‘COMMON-NAME’] = COMMON-NAME TYPES (0-n) => for each, check or create new node class, create rlt (node is_a_class types) SYNONYMS (0-n) => for each, create new node name, create rlt (node has_name synonyms)

compounds.dat: for each compound: create new node / class = compound UNIQUE-ID (1) => node.id = UNIQUE-ID COMMON-NAME (0-n) => node.Misc['COMMON-NAME'] = COMMON-NAME INCHI-KEY (0-1) {InChIKey=XXX} => node.misc['INCHI_KEY': XXX] MOLECULAR-WEIGHT (0-1) => node.misc()['MOLECULAR_WEIGHT'] = MOLECULAR-WEIGHT SMILES (0-1) => node.misc()['SMILES'] = SMILES TYPES (0-n) => for each, check or create new node class, create rlt (node is_a_class types) SYNONYMS (0-n) => for each, create new node name, create rlt (node has_name name) DBLINKS (0-n) {(db "id" ...)} => for each, create new node xref, create rlt (node has_xref xref)

proteins.dat: for each protein: create new node / class = protein UNIQUE-ID (1) => node.id = UNIQUE-ID COMMON-NAME (0-n) => node.Misc['COMMON-NAME'] = COMMON-NAME INCHI-KEY (0-1) {InChIKey=XXX} => node.misc['INCHI_KEY': XXX] MOLECULAR-WEIGHT (0-1) => node.misc()['MOLECULAR_WEIGHT'] = MOLECULAR-WEIGHT SMILES (0-1) => node.misc()['SMILES'] = SMILES TYPES (0-n) => for each, check or create new node class, create rlt (node is_a_class types) SYNONYMS (0-n) => for each, create new node name, create rlt (node has_name name) DBLINKS (0-n) {(db "id" ...)} => for each, create new node xref, create rlt (node has_xref xref) SPECIES (0-1) => for each, check or create new node class, create rlt (node is_in_species class)

reactions.dat: for each reaction: create new node / class = reaction + node.misc()["DIRECTION"] = "UNKNOWN" by default UNIQUE-ID (1) => node.id = UNIQUE-ID COMMON-NAME (0-n) => node.Misc['COMMON-NAME'] = COMMON-NAME EC-NUMBER (0-n) => node.Misc['EC-NUMBER'] = EC-NUMBER REACTION-DIRECTION (0-1) => node.Misc['DIRECTION'] = reaction-direction, if REVERSIBLE, else: LEFT-TO-RIGHT RXN-LOCATIONS (0,n) => node.misc['COMPARTMENT'] = rxn-location TYPES (0-n) => check or create new node class, create rlt (node.id is_a_class types's_node.id) DBLINKS (0-n) {(db "id" ...)} => create new node xref, create rlt (node has_xref xref's_node.id) SYNONYMS (0-n) => create new node name, create rlt (node has_name name's_node.id) – for LEFT and RIGHT, also check 2 next lines if info about 'coefficient' or 'compartment' defaut value: coefficient/stoichiometry = 1, compartment = unknown also check if the direction is 'RIGHT-TO-LEFT', if yes, inverse consumes and produces relations then change direction to 'LEFT-TO-RIGHT' LEFT (1-n) => create rlt (node.id consumes left's_node.id) RIGHT (1-n) => create rlt (node.id produces right's_node.id)

enzrxns.dat: for each association enzyme/reaction: create new rlt / type = catalyses ENZYME (1) => stock enzyme as 'enzyme catalyses' REACTION (1-n) => for each reaction after, create relation 'enzyme catalyses reaction'

pathways.dat: for each pathway: create new node / class = pathway UNIQUE-ID (1) => node._id = UNIQUE-ID TYPES (0-n) => check or create new node class, create rlt (node is_a_class types) COMMON-NAME (0-n) => node.Misc['COMMON-NAME'] = COMMON-NAME DBLINKS (0-n) {(db "id" ...)} => create new node xref, create rlt (node has_xref xref) SYNONYMS (0-n) => create new node name, create rlt (node has_name name) IN-PATHWAY (0-n) => check or create new node pathway, create rlt (node is_in_pathway name) REACTION-LIST (0-n) => check or create new node pathway, create rlt (node is_in_pathway name)

```
usage:
    pgdb_to_padmet.py --pgdb=DIR --output=FILE [--version=V] [--db=ID] [--
↪padmetRef=FILE] [--source=STR] [-v] [--enhance]
    pgdb_to_padmet.py --pgdb=DIR --output=FILE --extract-gene [--no-orphan]  [--
↪version=V] [--db=ID] [--padmetRef=FILE] [--source=STR] [-v] [--enhance]

options:
    -h --help     Show help.
    --version=V    Xcyc version [default: N.A].
    --db=ID    Biocyc database corresponding to the pgdb (metacyc, ecocyc, ...)␣
↪[default: N.A].
```

(continues on next page)

(continued from previous page)

```
    --output=FILE    padmet file corresponding to the DB.
    --pgdb=DIR    directory containg all the .dat files of metacyc (data).
    --padmetRef=FILE    padmet of reference.
    --source=STR    Tag associated to the source of the reactions, used to ensure␣
↪traceability [default: GENOME].
    --enhance    use the metabolic-reactions.xml file to enhance the database.
    --extract-gene    use the genes_file (use if its a specie's pgdb, if metacyc, do␣
↪not use).
    --no-orhpan    use the genes_file (use if its a specie's pgdb, if metacyc, do not␣
↪use).
    -v   print info.
```

padmet_utils.connection.pgdb_to_padmet.**main**()

sbmlGenerator

Description: The module sbmlGenerator contains functions to generate sbml files from padmet and txt usign the libsbml package

```
usage:
    sbmlGenerator.py --padmet=FILE --output=FILE --sbml_lvl=STR [--model_id=STR] [--
↪obj_fct=STR] [--mnx_chem_prop=FILE] [--mnx_chem_xref=FILE] [-v]
    sbmlGenerator.py --padmet=FILE --output=FILE [--init_source=STR] [-v]
    sbmlGenerator.py --compound=FILE --output=FILE [--padmetRef=FILE] [-v]
    sbmlGenerator.py --reaction=FILE --output=FILE --padmetRef=FILE [-v]

option:
    -h --help    Show help.
    --padmet=FILE    path of the padmet file to convert into sbml
    --output=FILE    path of the sbml file to generate.
    --mnx_chem_prop=FILE    path of the MNX chemical compounds properties.
    --mnx_chem_xref=FILE    path of the mnx dict of chemical compounds id mapping.
    --reaction=FILE    path of file of reactions ids, one by line to convert to sbml.
    --compound=FILE    path of file of compounds ids, one by line to convert to sbml.
    --init_source=STR    Select the reactions of padmet to convert on sbml based on␣
↪the source of the reactions, check relations rxn has_reconstructionData.
    --sbml_lvl=STR    sbml level of output. [default 3]
    --obj_fct=STR    id of the reaction objective.
    -v   print info.
```

padmet_utils.connection.sbmlGenerator.**main**()

sbml_to_curation_form

Description: extract 1 reaction (if rxn_id) or a list of reactions (if rxn_file) from a sbml file to the form used in aureme for curation. For example use this script to extract specific missing reaction of a model to a just created metabolic network.

```
usage:
    sbml_to_curation_form.py --sbml=FILE --output=FILE --rxn_id=ID [--comment=STR] [--
↪extract-gene] [-v]
    sbml_to_curation_form.py --sbml=FILE --output=FILE --rxn_file=FILE [--
↪comment=STR] [--extract-gene] [-v]
```

(continues on next page)

(continued from previous page)

```
options:
    -h --help     Show help.
    --sbml=FILE    path of the sbml.
    --output=FILE    form containing the reaction extracted, form used for manual␣
↪curation in aureme.
    --rxn_id=FILE    id of one reaction to extract
    --rxn_file=FILE    file of reactions ids to extract, 1 id by line.
    --extract-gene    If true, extract also genes associated to reactions.
    --comment=STR    comment associated to the reactions in the form. Used to track␣
↪sources of curation in aureme [default: "N.A"].
    -v   print info
```

padmet_utils.connection.sbml_to_curation_form.**main**()

sbml_to_padmet

Description: There are 3 cases of convertion sbml to padmet:

1./ Creation of a reference database in padmet format from sbml(s) (or updating one with new(s) sbml(s)) First usage, padmetRef is the padmetRef to create or to update. If it’s an update case, the output can be used to create a new padmet, if output None, will overwritte the input padmetRef.

2./ Creation of a padmet representing an organism in padmet format from sbml(s) (or updating one with new(s) sbml(s)) 2.A/ Without a database of reference: Second usage, padmetSpec is the padmetSpec to create or update. If it’s an update case, the output can be used to create a new padmet, if output None, will overwritte the input padmetSpec.

2.B/ With a database of refence: Third usage, padmetSpec is the padmetSpec to create or update. If it’s an update case, the output can be used to create a new padmet, if output None, will overwritte the input padmetSpec. padmetRef is the padmet representing the database of reference.

It is possible to define a specific policy and info for the padmet. To learn more about policy and info check doc of lib.padmetRef/Spec. if the ids of reactions/compounds are not the same between padmetRef and the sbml, it is possible to use a dictionnary of association (sbml_id padmetRef_id) with one line = ‘id_sbml id_padmetRef’ Finally if a reaction from sbml is not in padmetRef, it is possible to force the copy and creating a new reaction in padmetSpec with the arg -f

```
usage:
    sbml_to_padmet.py --sbml=DIR/FILE --padmetRef=FILE [--output=FILE] [--db=STR] [--
↪version=STR] [-v]
    sbml_to_padmet.py --sbml=DIR/FILE --padmetSpec=FILE [--output=FILE] [--source_
↪tool=STR] [--source_category=STR] [--source_id=STR] [-v]
    sbml_to_padmet.py --sbml=DIR/FILE --padmetSpec=FILE  --padmetRef=FILE  [--
↪mapping=DIR/FILE] [--mapping_tag=STR] [--output=FILE] [--source_tool=STR] [--source_
↪category=STR] [--source_id=STR] [-v] [-f]

options:
    -h --help     Show help.
    --padmetSpec=FILE    path to the padmet file to update with the sbml. If there's␣
↪no padmetSpec, just specify the output
    --padmetRef=FILE    path to the padmet file representing to the database of␣
↪reference (ex: metacyc_18.5.padmet)
    --sbml=FILE    1 sbml file to convert into padmetSpec (ex: my_network.xml/sbml)␣
↪OR a directory with n SBML
    --output=FILE   pathanme to the new padmet file
    --mapping=FILE    dictionnary of association id_origin id_ref
```

(continues on next page)

(continued from previous page)

```
    --mapping_tag=STR    if sbml is a folder, use a tag to define mapping files ex:␣
↪org1.sbml and org1_dict.csv, '_dict.csv' will be the mapping tag. [default: _dict.
↪csv]
    --db=STR    database name
    --version=STR    database version
    -v   print info
```

padmet_utils.connection.sbml_to_padmet.**main**()

wikiGenerator

Description: Contains all necessary functions to generate wikiPages from a padmet file and update a wiki online. Require WikiManager module (with wikiMate,Vendor)

```
usage:
    wikiGenerator.py --padmet=FILE/DIR --output=DIR --wiki_id=STR [--database=STR] [--
↪padmetRef=FILE] [--log_file=FILE] [-v]
    wikiGenerator.py --aureme_run=DIR --padmetSpec=ID -v

options:
    -h --help     Show help.
    --padmet=FILE    path to padmet file.
    --output=DIR    path to folder to create with all wikipages in subdir.
    --wiki_id=STR    id of the wiki.
    --padmetRef=FILE    path to padmet of reference, ex: metacyc_xx.padmet, if given,␣
↪able to calcul pathway rate completion.
    --log_file=FILE    log file from an aureme run, use this file to create a␣
↪wikipage with all the command used during the aureme run.
    --aureme_run=DIR    can use an aureme run as input, will use from config file␣
↪information for model_id and log_file and padmetRef.
    -v    print info.
```

padmet_utils.connection.wikiGenerator.**main**()

4.1.2 Scripts: Exploration

Description:

#TODO

compare_padmet

Description: #Compare 1-n padmet and create a folder output with files: genes.csv:

fieldnames = [gene, padmet_a, padmet_b, padmet_a_rxn_assoc, padmet_b_rxn_assoc] line = [gene-a, 'present' (if in padmet_a), 'present' (if in padmet_b), rxn-1;rxn-2 (names of reactions associated to gene-a in padmet_a), rxn-2]

reactions.csv: fieldnames = [reaction, padmet_a, padmet_b, padmet_a_genes_assoc, padmet_b_genes_assoc, padmet_a_formula, padmet_b_formula] line = [rxn-1, 'present' (if in padmet_a), 'present' (if in padmet_b), 'gene-a;gene-b; gene-a, 'cpd-1 + cpd-2 => cpd-3', 'cpd-1 + cpd-2 => cpd-3']

pathways.csv: fieldnames = [pathway, padmet_a_completion_rate, padmet_b_completion_rate, padmet_a_rxn_assoc, padmet_b_rxn_assoc] line = [pwy-a, 0.80, 0.30, rxn-a;rxn-b; rxn-a]

compounds.csv: fieldnames = [‘metabolite’, padmet_a_rxn_consume, padmet_a_rxn_produce, padmet_b_rxn_consume, padmet_rxn_produce] line = [cpd-1, rxn-1,’ ‘,rxn-1,’ ‘]

```
usage:
    compare_padmet.py --padmet=FILES/DIR --output=DIR [--padmetRef=FILE] [-v]

option:
    -h --help    Show help.
    --padmet=FILES/DIR    pathname of the padmet files, sep all files by ',', ex: /
↪path/padmet1.padmet;/path/padmet2.padmet OR a folder
    --output=DIR    pathname of the output folder
    --padmetRef=FILE    pathanme of the database ref in padmet
```

`padmet_utils.exploration.compare_padmet.main()`

compare_sbml

Description: compare reactions in two sbml.

Returns if a reaction is missing

And if a reaction with the same id is using different species or different reversibility

```
usage:
    compare_sbml.py --sbml1=FILE --sbml2=FILE

option:
    -h --help    Show help.
    --sbml1=FILE    path of the first sbml file
    --sbml2=FILE    path of the second sbml file
```

compare_sbml_padmet

Description: compare reactions in sbml and padmet file

```
usage:
    compare_sbml_padmet.py --padmet=FILE --sbml=FILE

option:
    -h --help    Show help.
    --padmet=FILE    path of the padmet file
    --sbml=FILE    path of the sbml file
```

`padmet_utils.exploration.compare_sbml_padmet.main()`

convert_sbml_db

Description: This tool is use the MetaNetX database to check or convert a sbml. Flat files from MetaNetx are required to run this tool. They can be found in the aureme workflow or from the MetaNetx website. To use the tool set:

mnx_folder= the path to a folder containing MetaNetx flat files. the files must be named as ‘reac_xref.tsv’ and ‘chem_xref.tsv’ or set manually the different path of the flat files with:

mnx_reac= path to the flat file for reactions

mnx_chem= path to the flat file for chemical compounds (species)

To check the database used in a sbml:

to check all element of sbml (reaction and species) set: to–map=all

to check only reaction of sbml set: to–map=reaction

to check only species of sbml set: to–map=species

To map a sbml and obtain a file of mapping ids to a given database set:

to-map: as previously explained

db_out: the name of the database target: ['metacyc', 'bigg', 'kegg'] only

output: the path to the output file

For a given sbml using a specific database.

Return a dictionnary of mapping.

the output is a file with line = reaction_id/or species in sbml, reaction_id/species in db_out database

ex: For a sbml based on kegg database, db_out=metacyc: the output file will contains for ex:

R02283 ACETYLORNTRANSAM-RXN

```
usage:
    convert_sbml_db.py --mnx_reac=FILE --mnx_chem=FILE --sbml=FILE --to-map=STR [-v]
    convert_sbml_db.py --mnx_folder=DIR --sbml=FILE --to-map=STR [-v]
    convert_sbml_db.py --mnx_folder=DIR --sbml=FILE --output=FILE --db_out=ID --to-
↪map=STR [-v]
    convert_sbml_db.py --mnx_reac=FILE --mnx_chem=FILE --sbml=FILE --output=FILE --db_
↪out=ID --to-map=STR [-v]

options:
    -h --help     Show help.
    --to-map=STR     select the part of the sbml to check or convert, must be in ['all
↪', 'reaction', 'species']
    --mnx_reac=FILE     path to the MetaNetX file for reactions
    --mnx_chem=FILE     path to the MetaNetX file for compounds
    --sbml=FILE     path to the sbml file to convert
    --output=FILE     path to the file containing the mapping, sep = "       "
    --db_out=FILE     id of the output database in ["BIGG","METACYC","KEGG"]
    -v     verbose.
```

padmet_utils.exploration.convert_sbml_db.**main**()

dendrogram_reactions_distance

Description: Use reactions.csv file from compare_padmet.py to create a dendrogram using a Jaccard distance.

From the matrix absence/presence of reactions in different species computes a Jaccard distance between these species. Apply a hierarchical clustering on these data with a complete linkage. Then create a dendrogram. Apply also intervene to create an upset graph on the data.

```
usage:
    dendrogram_reactions_distance.py --reactions=FILE --output=FILE [--padmetRef=STR]␣
↪[--pvclust] [--upset=INT] [-v]

option:
    -h --help    Show help.
```

(continues on next page)

(continued from previous page)

```
    -r --reactions=FILE    pathname of the file containing reactions in each species␣
↪of the comparison.
    -o --output=FOLDER    path to the output folder.
    --pvclust    launch pvclust dendrogram using R
    --padmetRef=STR    path to the padmet Ref file
    -u --upset=INT    number of cluster in the upset graph.
    -v    verbose mode.
```

flux_analysis

Description: Run flux balance analyse with cobra package. If the flux is >0. Run also FVA and return result in standard output

```
usage:
    flux_analysis.py --sbml=FILE
    flux_analysis.py --sbml=FILE --seeds=FILE --targets=FILE
    flux_analysis.py --sbml=FILE --all_species


option:
    -h --help    Show help.
    --sbml=FILE    pathname to the sbml file to test for fba and fva.
    --seeds=FILE    pathname to the sbml file containing the seeds (medium).
    --targets=FILE    pathname to the sbml file containing the targets.
    --all_species    allow to make FBA on all the metabolites of the given model.
```

get_pwy_from_rxn

Description: From a file containing a list of reaction, return the pathways where these reactions are involved. ex: if rxn-a in pwy-x => return, pwy-x; all rxn ids in pwy-x; all rxn ids in pwy-x FROM the list; ratio

```
usage:
    get_pwy_from_rxn.py --reaction_file=FILE --padmetRef=FILE  --output=FILE

options:
    -h --help     Show help.
    --reaction_file=FILE    pathname of the file containing the reactions id, 1/line
    --padmetRef=FILE    pathname of the padmet representing the database.
    --output=FILE    pathname of the file with line = pathway id, all reactions id,␣
↪reactions ids from reaction file, ratio. sep = "         "
```

`padmet_utils.exploration.get_pwy_from_rxn.`**`main`**`()`

padmet_stats

Description: From a file containing a list of reaction, return the pathways where these reactions are involved. ex: if rxn-a in pwy-x => return, pwy-x; all rxn ids in pwy-x; all rxn ids in pwy-x FROM the list; ratio

```
usage:
    get_pwy_from_rxn.py --reaction_file=FILE --padmetRef=FILE  --output=FILE

options:
```

(continues on next page)

(continued from previous page)

```
    -h --help     Show help.
    --reaction_file=FILE    pathname of the file containing the reactions id, 1/line
    --padmetRef=FILE    pathname of the padmet representing the database.
    --output=FILE    pathname of the file with line = pathway id, all reactions id,␣
↪reactions ids from reaction file, ratio. sep = "         "
```

padmet_utils.exploration.get_pwy_from_rxn.**main**()

report_network

Description: Create reports of a padmet file.

> all_pathways.tsv: header = ["dbRef_id", "Common name", "Number of reaction found", "Total number of reaction", "Ratio (Reaction found / Total)"]
>
> all_reactions.tsv: header = ["dbRef_id", "Common name", "formula (with id)", "formula (with common name)", "in pathways", "associated genes"]
>
> all_metabolites.tsv: header = ["dbRef_id", "Common name", "Produced (p), Consumed (c), Both (cp)"]

```
usage:
    report_network.py --padmetSpec=FILE --output_dir=dir [--padmetRef=FILE] [-v]

options:
    -h --help     Show help.
    --padmetSpec=FILE    pathname of the padmet file.
    --padmetRef=FILE    pathname of the padmet file used as database
    --output_dir=dir    directory for the results.
    -v   print info.
```

padmet_utils.exploration.report_network.**main**()

visu_path

Description: #TODO: not stable version Allows to visualize a pathway in padmet network.

Color code: reactions associated to the pathway, present in the network: lightGreen reactions associated to the pathway, not present in the network: red compounds: skyblue

```
usage:
    visu_path.py --padmetSpec=FILE --padmetRef=FILE --pathway=ID

options:
    -h --help     Show help.
    --padmetSpec=FILE    pathname to the PADMet file of the network.
    --padmetRef=FILE    pathname to the PADMet file of the db of reference.
    --pathway=ID    pathway id to visualize.
```

padmet_utils.exploration.visu_path.**main**()

padmet_utils.exploration.visu_path.**save_plot**(*plot*, *filepath*)

Saves plot in multiple formats

Parameters

- **arg1** (*<plot object>*) – plot object
- **arg2** (*<str>*) – filename with its path

4.1.3 Scripts: Management

Description:

#TODO

manual_curation

Description: Update a padmetSpec by filling specific forms.

1./ Create new reaction(s) to padmet file.

- Get the template form with –template_new_rxn
- Fill the template
- set –data as path to the filled template

2./ Add reaction(s) from padmetRef or remove reactions(s).

- Get the template form with –template_add_delete_rxn
- Fill the template
- set –date as path to the filled template

Update padmetSpec and create a new padmet (new_padmet) or overwrite the input

```
usage:
    manual_curation.py --padmetSpec=FILE --data=FILE [--padmetRef=FILE] [--
↪output=FILE] [--tool=STR] [--category=STR] [-v]
    manual_curation.py --template_new_rxn=FILE
    manual_curation.py --template_add_delete_rxn=FILE

option:
    -h --help    Show help.
    --padmetSpec=FILE    path to the padmet to update
    --padmetRef=FILE    path of the padmet representing the reference database
    --data=FILE    path to the form with data for curation
    --output=FILE    path to the output. if None. Overwriting padmetSpec
    --tool=STR    specification of the tool used to allow this curation: ex a tool of␣
↪gapfilling (meneco)
    --category=STR    specification of the category of curation: ex if a reaction is␣
↪added based on annotation info, use 'annotation'
    --template_new_rxn=FILE    create a form used to create new reaction, use this␣
↪form as input for 'data' option
    --template_add_delete_rxn=FILE    create a form used to add or delete reaction,␣
↪use this form as input for 'data' option
    -v    print info
```

padmet_utils.management.manual_curation.**main**()

padmet_compart

Description: For a given padmet file, check and update compartment.

1./ Get all compartment with 1st usage

2./ Remove a compartment with 2nd usage. Remove all reactions acting in the given compartment

3./ change compartment id with 3rd usage

```
usage:
    padmet_compart.py --padmet=FILE
    padmet_compart.py --padmet=FILE --remove=STR [--output=FILE] [-v]
    padmet_compart.py --padmet=FILE --old=STR --new=STR [--output=FILE] [-v]

options:
    -h --help     Show help.
    --padmet=FILE    pathname of the padmet file
    --remove=STR    compartment id to remove
    --old=STR    compartment id to change to new id
    --new=STR    new compartment id
    --output=FILE    new padmet pathname, if none, overwritting the original padmet
    -v   print info
```

`padmet_utils.management.padmet_compart.`**`main`**`()`

padmet_medium

Description: For a given set of compounds representing the growth medium (or seeds). Create 2 reactions for each compounds to maintain consistency of the network for flux analysis. For each compounds create:

An exchange reaction: this reaction consumes the compound in the compartment 'C-BOUNDARY' and produces the compound in the compartment 'e' extracellular

A transport reaction: this reaction consumes the compound in the compartment 'e' extracellular' and produces the compound in the compartment 'c' cytosol ex: for seed 'cpd-a'

1/ check if cpd-a in padmetSpec, if not, copy from padmetRef.

2/ create exchange reaction: ExchangeSeed_cpd-a_b: 1 cpd-a (C-BOUNDARAY) <=> 1 cpd-a (e)

3/ create transport reaction: TransportSeed_cpd-a_e: 1 cpd-a (e) => 1 cpd-a (c)

4/ create a new file if output not None, or overwrite padmetSpec

```
usage:
    padmet_medium.py --padmetSpec=FILE
    padmet_medium.py --padmetSpec=FILE -r [--output=FILE] [-v]
    padmet_medium.py --padmetSpec=FILE --seeds=FILE [--padmetRef=FILE] [--
↪output=FILE] [-v]

options:
    -h --help     Show help.
    --padmetSpec=FILE    path to the padmet file to update
    --padmetRef=FILE    path to the padmet file representing to the database of␣
↪reference (ex: metacyc_18.5.padmet)
    --seeds=FILE    the path to the file containing the compounds ids and the compart,
↪ line = cpd-id        compart.
    --output=FILE    If not None, pathname to the padmet file updated
    -r    Use to remove all medium from padmet
    -v   print info
```

`padmet_utils.management.padmet_medium.`**`main`**`()`

Python Module Index

Index