sygma Documentation

Release 1.1.0

Lars Ridder

Mar 19, 2018

Contents

Contents:

CHAPTER 1

API

class scenario.**Rule**(*rulename*, *probability*, *smarts*)
Class to contain a metabolic rule

Parameters

- **rulename** – A string containing a unique name of the rule
- **probability** – A probability value between 0 and 1 indicating the empirical success rate of the rule
- **smarts** – A reaction smarts describing the chemical transformation of the rule

class scenario.**Scenario**(*scenario*)
Class to read and process metabolic scenario

Parameters **scenario** – A list of lists, each representing a metabolic phase as [name_of_file_containing_rules, number_of_cycles_to_apply]

run(*parentmol*)

Parameters **parentmol** – An RDKit molecule

Returns A sygma.Tree object

class tree.**Tree**(*parentmol=None*)
Class to build and analyse a metabolic tree

Parameters **parentmol** – An RDKit molecule

add_coordinates()
Add missing atomic coordinates to all metabolites

calc_scores()
Calculate probability scores for all metabolites

metabolize_all_nodes(*rules*, *cycles=1*)
Metabolize all nodes according to [rules], for [cycles] number of cycles

Parameters

- **rules** – List of rules
- **cycles** – Integer indicating the number of subsequent steps to apply the rules

to_list (*filter_small_fragments=True*, *parent_column='parent'*)
Generate a list of metabolites

Parameters

- **filter_small_fragments** – Boolean to activate filtering all metabolites with less then 15% of original atoms (of the parent)
- **parent_column** – String containing the name for the column with the parent molecule

Returns A list of dictionaries for each metabolites, containing the SyGMa_metabolite (an RDKit Molecule), SyGMa_pathway and SyGMa_score, sorted by decreasing probability.

to_smiles (*filter_small_fragments=True*)
Generate a smiles list of metabolites

Parameters **filter_small_fragments** – Boolean to activate filtering all metabolites with less then 15% of original atoms (of the parent)

Returns A list of metabolites as list `[[SyGMa_metabolite as smiles, SyGMa_score]]` sorted by decreasing probability score.

write_sdf (*file=<open file '<stdout>'*, *mode 'w'>*, *filter_small_fragments=True*)
Generate an SDFile with metabolites including the SyGMa_pathway and the SyGMa score as properties

Parameters

- **file** – The SDF file to write to
- **filter_small_fragments** – Boolean to activate filtering all metabolites with less then 15% of original atoms (of the parent)

class treenode.**TreeNode** (*mol*, *parent=''*, *rule=None*, *score=None*, *pathway=''*)
Class containing a node of the SyGMa tree

Key mol RDKit Molecule

Key parents Dictonary {inchikey_of_parent: rulename_transforming_parent_to_self}

Key children List of inchikeys of the child nodes

Key score Value between 0 and 1

Key pathway String describing the pathway from parent to self

Key n_original_atoms Integer, number of atoms originating from parent or None if not yet determined

gen_coords ()
Calculate 2D positions for atoms in self.mol without coordinates

CHAPTER 2

Command line script

SyGMa: Systematically Generating potential Metabolites

```
usage: sygma [-h] [--version] [-o OUTPUTTYPE] [-1 PHASE1] [-2 PHASE2]
             [-l {debug,info,warn, error}]
             parentmol
```

2.1 Positional Arguments

parentmol Smiles string of parent molecule structure

2.2 Named Arguments

--version show program's version number and exit

-o, --outputtype Molecule output type (default: sdf)

Default: sdf

-1, --phase1 Number of phase 1 cycles (default: 1)

Default: 1

-2, --phase2 Number of phase 2 cycles (default: 1)

Default: 1

-l, --loglevel Possible choices: debug, info, warn, error

Set logging level (default: "info")

Default: "info"

CHAPTER 3

Introduction

SyGMa is a python library for the **Sy**stematic **G**eneration of potential **M**eta**a**bolites. It is a reimplementation of the metabolic rules outlined in Ridder, L., & Wagener, M. (2008) SyGMa: combining expert knowledge and empirical scoring in the prediction of metabolites. ChemMedChem, 3(5), 821-832.

3.1 Requirements

SyGMa requires RDKit with INCHI support

3.2 Installation

- Install with Anaconda: `conda install -c 3d-e-Chem -c rdkit sygma`

OR

- Install RDKit following the instructions in http://www.rdkit.org/docs/Install.html

AND

- `pip install sygma` OR, after downloading sygma, `python setup.py install`

3.3 Example

```
import sygma
from rdkit import Chem


def test_predict_phenol_metabolites():
    """Test prediction of phenol metabolites by sygma module"""
```

(continues on next page)

(continued from previous page)

```
    # Each step in a scenario lists the ruleset and the number of reaction cycles to␣
↪be applied
    scenario = sygma.Scenario([
        [sygma.ruleset['phase1'], 1],
        [sygma.ruleset['phase2'], 1]])

    # An rdkit molecule, optionally with 2D coordinates, is required as parent␣
↪molecule
    parent = Chem.MolFromSmiles("c1ccccc1O")

    metabolic_tree = scenario.run(parent)
    metabolic_tree.calc_scores()

    metabolite_list = metabolic_tree.to_list()
    assert len(metabolite_list) == 12
    assert metabolite_list[0]['SyGMa_score'] == 1
    assert metabolite_list[1]['SyGMa_pathway'] == 'O-glucuronidation_(aromatic_
↪hydroxyl); \n'
```

3.4 Docker

SyGMa can be executed in a Docker container as follows:

```
docker run 3dechem/sygma c1ccccc1O
```

3.5 Rulesets

SyGMa comes currently with two rulesets:

phase1 Phase 1 metabolism rules include mainly different types of oxidation, hydrolysis, reduction and condensation reactions

phase2 Phase 2 metabolism rules include severaly conjugation reaction, i.e. with glucuronyl, sulfate, methyl and acetyl

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index