moiety_modeling Documentation

Release 1.0.3.2

Huan Jin, Hunter N.B. Moseley

Sep 30, 2019

Contents:

CHAPTER 1

User Guide

1.1 Description

The *moiety_modeling* package provides a simple Python interface for moiety model representation, model optimization and model selection. Both moiety models and isotopologue datasets are stored in'JSON' files, which can be used for further model optimization, selection, analysis and visualization.

1.2 Installation

moiety_modeling runs under Python 3.6+ and is available through python3-pip. Install via pip or clone the git repo and install the following dependencies and you are ready to go!

1.2.1 Install on Linux

Pip installation (method 1)

```
python3 -m pip install moiety-modeling
```

GitHub Package installation (method 2)

Make sure you have git installed:

```
git clone https://github.com/MoseleyBioinformaticsLab/moiety_modeling.git
```

Dependecies

moiety_modeling requires the following Python libraries:

- docopt for creating the command-line interface.
- jsonpickle for saving Python objects in a JSON serializable form and outputting to a file.
- numpy and matplotlib for visualization of optimized results.
- scipy for application of optimization methods.
- SAGA-optimize for parameters optimization.

1.3 Basic usage

The `moiety_modeling` package can be used in several ways:

- As a library for accessing and manipulating moiety models and isotopologue datasets stored in the *JSON* files.
- **As a command-line tool:**
  - Optimize the moiety model parameters.
  - Analyze the optimization results of moiety model, and select the optimal model.
  - Visuslize the optimized results.

Note: Read *The moiety_modeling Tutorial* to learn more and see code examples on using the `moiety_modeling` as a library and as a command-line tool.

CHAPTER 2

The moiety_modeling Tutorial

The *moiety_modeling* can be used to:

- Construct a moiety model.
- Optimize parameters of a moiety model.
- Analyze optimized results and select the optimal model.
- Visualize the optimized results.

In this document, each use will be explained in details.

2.1 The moiety_modeling API tutorial

2.1.1 Using moiety_modeling to construct a moiety model

In moiety modeling, we dissemble molelcules into molecular parts called moieties (called functional groups in chemistry), and then deconvolute molecular isotopic incorporation into moiety isotopic. A constructed moiety model is then stored in a JSON file.

Here is an example of moiety model construction.

```
>>> import moiety_modeling
>>> import jsonpickle
>>> ribose = moiety_modeling.Moiety("ribose", {'13C': 5}, isotopeStates={'13C': [0, 2,
↪ 3, 5]}, nickname='r')    # moiety creation
>>> glucose = moiety_modeling.Moiety("glucose", {'13C': 6}, isotopeStates={'13C': [0,␣
↪3, 6]}, nickname='g')
>>> acetyl = moiety_modeling.Moiety("acetyl", {'13C': 2}, isotopeStates={'13C': [0,␣
↪2]}, nickname='a')
>>> uracil = moiety_modeling.Moiety("uracil", {'13C': 4}, isotopeStates={'13C': [0, 1,
↪ 2, 3]}, nickname='u')
>>> relationship1 = moiety_modeling.Relationship(glucose1, '13C3', ribose1, '13C2', '*
↪', 2)    # relationship creation
```

(continues on next page)

(continued from previous page)

```
>>> relationship2 = moiety_modeling.Relationship(ribose1, '13C3', ribose1, '13C2')
>>> relationship3 = moiety_modeling.Relationship(glucose1, '13C0', ribose1, '13C0')
>>> relationship4 = moiety_modeling.Relationship(glucose1, '13C6', ribose1, '13C5')
>>> UDP_GlcNAC = moiety_modeling.Molecule('UDP_GlcNAC', [ribose, glucose, acetyl,␣
↪uracil])   # molecule creation
>>> model = moiety_modeling.Model('6_G0R2A1U3_g3r2r3_g6r5', [ribose, glucose, acetyl,␣
↪uracil], [UDP_GlcNAC], [relationship1, relationship2, relationship3, relationship4])
>>> with open('model.json', 'w') as outFile:   # store the model into JSONPickle file.
        outFile.write(jsonpickle.encode({'models': [model]}))
```

The moiety states can also be constructed using all the possible states.

```
>>> glucose = moiety_modeling.Moiety("glucose", {'13C': 6, '18O': 5}, states=['13C_0.
↪18O_0', '13C_6.18O_5'], nickname='g')
>>> ribose = moiety_modeling.Moiety("ribose", {'13C': 5, '18O': 4}, states=['13C_0.
↪18O_0', '13C_5.18O_4'], nickname='r')
>>> acetyl = moiety_modeling.Moiety("acetyl", {'13C': 2, '18O': 1}, states=['13C_0.
↪18O_0', '13C_2.18O_1'], nickname='a')
>>> uracil = moiety_modeling.Moiety("uracil", {'13C': 4, '18O': 2}, states=['13C_0.
↪18O_0', '13C_1.18O_0', '13C_2.18O_1', '13C_2.18O_0', '13C_3.18O_0', '13C_3.18O_1'],␣
↪nickname='u')
```

2.1.2 Using moiety_modeling to store isotopologue dataset

The Dataset class in the 'moiety_modeling' package organizes a single mass spectroscopy isotopologue profile dataset into a dictionary-based data structure.

Here is an example of dataset construction.

```
>>> import moiety_modeling
>>> import jsonpickle
>>> dataset1 = moiety_modeling.Dataset("12h", {'UDP_GlcNAC':
              [{'labelingIsotopes': '13C_0', 'height': 0.0175442549, 'heightSE': 0},
               {'labelingIsotopes': '13C_1', 'height': 0, 'heightSE': 0},
               {'labelingIsotopes': '13C_2', 'height': 0.0007113347, 'heightSE': 0},
               {'labelingIsotopes': '13C_3', 'height': 0.0002990498, 'heightSE': 0},
               {'labelingIsotopes': '13C_4', 'height': 0.0012322448, 'heightSE': 0},
               {'labelingIsotopes': '13C_5', 'height': 0.0962990868, 'heightSE': 0},
               {'labelingIsotopes': '13C_6', 'height': 0.0737941503, 'heightSE': 0},
               {'labelingIsotopes': '13C_7', 'height': 0.0194440036, 'heightSE': 0},
               {'labelingIsotopes': '13C_8', 'height': 0.063026207, 'heightSE': 0},
               {'labelingIsotopes': '13C_9', 'height': 0.0058731399, 'heightSE': 0},
               {'labelingIsotopes': '13C_10', 'height': 0.0312896069, 'heightSE': 0},
               {'labelingIsotopes': '13C_11', 'height': 0.3124695022, 'heightSE': 0},
               {'labelingIsotopes': '13C_12', 'height': 0.0573898846, 'heightSE': 0},
               {'labelingIsotopes': '13C_13', 'height': 0.277122791, 'heightSE': 0},
               {'labelingIsotopes': '13C_14', 'height': 0.0234859781, 'heightSE': 0},
               {'labelingIsotopes': '13C_15', 'height': 0.0200187655, 'heightSE': 0},
               {'labelingIsotopes': '13C_16', 'height': 0, 'heightSE': 0},
               {'labelingIsotopes': '13C_17', 'height': 0, 'heightSE': 0}]})
>>> dataset2 = moiety_modeling.Dataset("24h", {'UDP_GlcNAC':
              [{'labelingIsotopes': '13C_0', 'height': 0.00697626, 'heightSE': 0},
               {'labelingIsotopes': '13C_1', 'height': 0, 'heightSE': 0},
               {'labelingIsotopes': '13C_2', 'height': 0.0008426934, 'heightSE': 0},
               {'labelingIsotopes': '13C_3', 'height': 0.0007070956, 'heightSE': 0},
```

(continues on next page)

(continued from previous page)

```
                {'labelingIsotopes': '13C_4', 'height': 0.0006206594, 'heightSE': 0},
                {'labelingIsotopes': '13C_5', 'height': 0.068147345, 'heightSE': 0},
                {'labelingIsotopes': '13C_6', 'height': 0.0499393097, 'heightSE': 0},
                {'labelingIsotopes': '13C_7', 'height': 0.023993641, 'heightSE': 0},
                {'labelingIsotopes': '13C_8', 'height': 0.062901247, 'heightSE': 0},
                {'labelingIsotopes': '13C_9', 'height': 0.0056603032, 'heightSE': 0},
                {'labelingIsotopes': '13C_10', 'height': 0.0281210238, 'heightSE': 0},
                {'labelingIsotopes': '13C_11', 'height': 0.2482899264, 'heightSE': 0},
                {'labelingIsotopes': '13C_12', 'height': 0.0613088541, 'heightSE': 0},
                {'labelingIsotopes': '13C_13', 'height': 0.3325253653, 'heightSE': 0},
                {'labelingIsotopes': '13C_14', 'height': 0.0499904271, 'heightSE': 0},
                {'labelingIsotopes': '13C_15', 'height': 0.0537153908, 'heightSE': 0},
                {'labelingIsotopes': '13C_16', 'height': 0.0062604583, 'heightSE': 0},
                {'labelingIsotopes': '13C_17', 'height': 0, 'heightSE': 0}]})    #␣
↪dataset creation
>>> with open('dataset.json', 'w') as outFile: # store dataset into JSONPickle file.
        outFile.write(jsonpickle.encode({'datasets': [dataset1, dataset2]}))
```

2.1.3 Setting optimization parameters

The optimization parameters are stored in a JSON file. Several optimization methods, including SAGA and three scipy optimization methods ('TNC', 'SLSQP', 'L_BFGS_B'), are available in the package. When using the SAGA optimization method, optimization parameters for SAGA should be specified. The parameters for scipy optimization methods can also be modified. Please refer the correponding API for detailed information.

Here is the example of optimization parameters construction.

```
>>> import jsonpickle
>>> setting = {'SAGA': {'methodParameters': {'alpha': 1, 'crossoverRate': 0.05,
↪'mutationRate': 3, 'populationSize': 20,
                'startTemperature': 0.5, 'stepNumber': 500, 'temperatureStepSize': 100}
↪,
                'noPrintAllResults': 1, 'noPrintBestResults': 0, 'optimizationSetting
↪': 'SAGA_500'},
                'TNC': {'methodParameters': None, 'optimizationSetting': 'TNC'} }
>>> with open('optimizationSetting.json', 'w') as outFile:
        outFile.write(jsonpickle.encode({'optimizations': setting}))
```

2.2 The moiety_modeling CLI tutorial

2.2.1 Using moiety_modeling to optimize parameters of moiety model

To conduct the optimization, moiety model, datasets, and optimization settings should be provided. Please use the -h for more information of the option parameters.

```
python3 -m moiety_modeling modeling --models=<model_jsonfile> --datasets=<dataset_
↪jsonfile> --optimizations=<optimizationSetting_json> --repetition=100 --split --
↪multiprocess --energyFunction=logDifference
```

2.2.2 Using moiety_modeling to analyze optimized results and select the optimal model

The *moiety_modeling* package provides facilities to analyze the optimization results, select the optimal model, and compare the selection results under different optimization settings. Please refer to API for detailed information of option parameters.

```
python3 -m moiety_modeling analyze optimizations --a <optimizationPaths_txtfile>   #␣
↪To analyze the optimization results of multiple moiety models together.
python3 -m moiety_modeling analyze optimizations --s <optimzationResults_jsonfile>
↪# To analyze the optimization results of a single moiety model.
python3 -m moiety_modeling analyze rank <analysisPaths_txtfile> --rankCriteria=AICc
↪# To rank models according to the selection criteria.
python3 -m moiety_modeling analyze table <rankPaths_txtfile>   # To compare the␣
↪selection results under different optimizaton settings.
```

2.2.3 Using moiety_modeling to visualize the optimized results

The 'moiety_modeling' package provides facilities to visualize the optimization results.

```
python3 -m moiety_modeling plot moiety <analysisResults_jsonfile>   # To plot the␣
↪distribution of calculated moiety modeling parameters.
python3 -m moiety_modeling plot isotopologue <analysisResults_jsonfile>   # To plot␣
↪the comparison of calculated and observed isotopologue intensities.
```

CHAPTER 3

The moiety_modeling API Reference

Routines for working with moiety modeling.

This package includes the following modules:

modeling This module provides the *Dataset* class to organize a single mass spectroscopy profile dataset into a dictionary-based data structure, the *ModelOptimization* and derived classes for performing a single model optimization, and the *OptimizationManager* class to manage the optimization process of moiety modeling.

model This module provides the *Moiety*, *Molecule*, *Relationship*, and *Model* classes for representing a moiety model.

analysis This module provides several classes to analyze the optimization results, select the optimal model, and visualize the results. The ResultAnalysis class is responsible for generating general statistics from the optimization results. The *ModelRank* class selects the model that best reflects the observed isotopologue profile. The *ComparisonTable* class compares the optimal model selected under different optimization settings. The *PlotMoietyDistribution* class plots the distribution of moiety value of the moiety model. The *PlotIsotopologueIntensity* class plots comparison of the observed and the calculated isotopologue intensity.

cli This module provides a command-line interface for the moiety_modeling package.

3.1 moiety_modeling.modeling

This module provides the *Dataset* class to organize a single mass spectroscopy isotopologue profile dataset into a dictionary-based data structure, the *ModelOptimization* and derived classes for performing a single model optimization, and the *OptimizationManager* class to manage the optimization process of moiety modeling.

The optimization results are stored in the JSON file. A txt file containing all the paths to the generated JSON files is created to facilitate further model analysis.

class moiety_modeling.modeling.**Dataset**(*datasetName*, **args*, ***kwargs*)

Dataset class that stores a single mass spectroscopy isotopologue profile in the form of UserDict.

__init__(*datasetName*, **args*, ***kwargs*)

Dataset initializer.

Parameters **datasetName** (*str*) – the name of the dataset.

class moiety_modeling.modeling.**ModelOptimization**(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*)

The abstract *ModelOptimization* class.

__init__(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*)

ModelOptimization initializer.

Parameters

- **model** (*Model*) – a *Model* instance.
- **datasets** (*list*) – a list of *Dataset* instances.
- **path** (*str*) – the subdirectory path to save the optimization results.
- **methodParameters** (*dict*) – the parameters for optimization method.
- **optimizationSetting** (*str*) – abbreviated name for the optimization.
- **energyFunction** (*str*) – the energy function used in the optimization.

energyCalculation(*vector*)

Calculate the energy of the model.

Parameters **vector** (*list*) – a list of model parameter values.

Return double the energy.

absDifferenceEnergyFunction(*moietyStateValue*, *dataset*)

The absolute difference energy function. The absolute value between the observed and the calculated isotopologues: energy = sum(**|I<cal> - I<obs>|**).

Parameters

- **moietyStateValue** (*list*) – a list of values for the moietyStates.
- **dataset** (*Dataset*) – a *Dataset* instance.

Returns the energy of the model.

Return type double

logDifferenceEnergyFunction(*moietyStateValue*, *dataset*)

The log difference energy function. The difference between the log of observed and the calculated isotopologues: energy = sum(**|log(I<cal>) - log(I<obs>)|**)

Parameters

- **moietyStateValue** (*list*) – a list of values for the moietyStates.
- **dataset** (*Dataset*) – a *Dataset* instance.

Returns the energy of the model.

Return type double

squareDifferenceEnergyFunction(*moietyStateValue*, *dataset*)

The absolute difference energy function. The absolute value between the observed and the calculated isotopologues: energy = sum(**|I<cal> - I<obs>|**).

Parameters

- **moietyStateValue** (*list*) – a list of values for the moietyStates.
- **dataset** (*Dataset*) – a *Dataset* instance.

Returns the energy of the model.

Return type double

AICDifferenceEnergyFunction(*moietyStateValue*, *dataset*)
The absolute difference energy function. The absolute value between the observed and the calculated isotopologues: energy = sum(|I<cal> - I<obs>|).

Parameters

- **moietyStateValue** (*list*) – a list of values for the moietyStates.
- **dataset** (*Dataset*) – a *Dataset* instance.

Returns the energy of the model.

Return type double

optimizationScripts()
To save the optimization scripts of the optimization for check. :return: None :rtype: None

class moiety_modeling.modeling.**SAGAoptimization**(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*, *noPrintBestResults*, *noPrintAllResults*)
The *SAGAoptimization* class (for combined datasets) inherited from *ModelOptimization*.

__init__(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*, *noPrintBestResults*, *noPrintAllResults*)
SAGAoptimization initializer.

Parameters

- **model** (*Model*) – a *Model* instance.
- **datasets** (*list*) – a list of *Dataset* instances.
- **path** (*str*) – the path to save the optimization results.
- **methodParameters** (*dict*) – the parameters for optimization method.
- **optimizationSetting** (*str*) – abbreviated name for the optimization.
- **energyFunction** (*str*) – the energy function used in the optimization.
- **noPrintBestResults** (*int*) – not to save the all the best results of the optimization process.
- **noPrintAllResults** (*int*) – not to save the all the results of the optimization process.

bestResultsFile(*i*)
Open the file to record the best results during the optimization process.

Parameters **i** (*int*) – the ith optimization.

Returns the file handler.

Return type TextIOWrapper

allResultsFile(*i*)
Open the file to record the all optimization results.

Parameters **i** (*int*) – the ith optimization.

Returns the file handler.

Return type TextIOWrapper.

optimizeSingle(*i*)
To perform one optimization.

Parameters **i** (*int*) – the ith optimization.

Returns the best Guess from the optimization process.

Return type Guess.

creatSubdir()
To create subdirectories to store the optimization process.

Returns None.

Return type None.

class moiety_modeling.modeling.**SAGAseparateOptimization**(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*, *noPrintBestResults*, *noPrintAllResults*)
The *SAGAseparateOptimization* class (for split dataset) inherited from *SAGAoptimization*.

__init__(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*, *noPrintBestResults*, *noPrintAllResults*)
SAGAseparateOptimization initializer.

Parameters

- **model** (*Model*) – a *Model* instance.
- **datasets** (*list*) – a list of *Dataset* instances.
- **path** (*str*) – the path to save the optimization results.
- **methodParameters** (*dict*) – the parameters for optimization method.
- **optimizationSetting** (*str*) – abbreviated name for the optimization.
- **energyFunction** (*str*) – the energy function used in the optimization.
- **noPrintBestResults** (*int*) – not to save the all the best results of the optimization process.
- **noPrintAllResults** (*int*) – not to save the all the results of the optimization process.

optimizeSingle(*i*)
Perform one optimization.

Parameters **i** (*int*) – the ith optimization.

Returns the best Guess from the optimization process.

Return type Guess.

class moiety_modeling.modeling.**ScipyGuess**(*elements*, *energy*)
To convert optimization results to Guess instance.

__init__(*elements*, *energy*)
ScipyGuess initializer.

Parameters

- **elements** (*list*) – a list of values for model parameters.
- **energy** (*double*) – the energy of the Guess calculated from an energy function.

class moiety_modeling.modeling.**ScipyOptimization**(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*, *method*)

The *ScipyOptimization* class (for combined datasets) inherited from *ModelOptimization*.

__init__(*model*, *datasets*, *path*, *methodParameters*, *optimizationSetting*, *energyFunction*, *method*)

ScipyOptimization initializer.

Parameters

- **model** (*Model*) – the *Model* instance.
- **datasets** (*list*) – a list of *Dataset* instances.
- **path** (*str*) – the path to save the optimization results.
- **methodParameters** (*dict*) – the parameters for optimization method.
- **optimizationSetting** (*str*) – abbreviated name for the optimization.
- **energyFunction** (*str*) – the energy function used in the optimization.
- **method** (*str*) – the scipy optimization method.

optimizeSingle(*i*)

To perform one optimization.

Parameters **i** – the ith optimization.

Returns the best *ScipyGuess* from the optimization process.

Return type *ScipyGuess*.

class moiety_modeling.modeling.**ScipySeparateOptimization**(*model*, *datasets*, *path*, *methodParameters*, *energyFunction*, *optimizationSetting*, *method*)

The *ScipySeparateOptimization* class (for split dataset) inherited from *ScipyOptimization*.

__init__(*model*, *datasets*, *path*, *methodParameters*, *energyFunction*, *optimizationSetting*, *method*)

ScipySeparateOptimization initializer.

Parameters

- **model** (*Model*) – the *Model* instance.
- **datasets** (*list*) – a list of *Dataset* instances.
- **path** (*str*) – the path to save the optimization results.
- **methodParameters** (*dict*) – the parameters for optimization method.
- **energyFunction** (*str*) – the energy function used in the optimization.
- **optimizationSetting** (*str*) – abbreviated name for the optimization.
- **method** (*str*) – the scipy optimization method.

optimizeSingle(*i*)

To perform single optimization.

Parameters **i** (*int*) – the ith optimization.

Returns the best *ScipyGuess* from the optimization process.

Return type *ScipyGuess*.

class moiety_modeling.modeling.**OptimizationManager**(*models*, *datasets*, *optimizations*, *path*, *split=True*, *multiprocess=True*, *force=True*, *times=100*, *energyFunction='logDifference'*, *printOptimizationScript=True*)

OptimizationManager class manage the optimization process based on the optimization settings.

__init__(*models*, *datasets*, *optimizations*, *path*, *split=True*, *multiprocess=True*, *force=True*, *times=100*, *energyFunction='logDifference'*, *printOptimizationScript=True*)

OptimizationManager initializer.

Parameters

- **models** (*list*) – a list of *Moiety* instances.
- **datasets** (*list*) – a list of *Dataset* instances.
- **optimizations** (*dict*) – a list of optimization settings.
- **path** (*str*) – the path to store the optimization results.
- **split** (*True*) – to split the dataset or not.
- **multiprocess** (*True*) – to apply multiprocess or not.
- **force** (*True*) – to force optimization process or not.
- **times** (*int*) – the times of optimization for each moiety model.
- **energyFunction** (*str*) – the energy function used for optimization.
- **printOptimizationScript** (*True*) – to print out the optimization script or not.

optimizeModels()

To optimize moiety models based on the optimization settings.

Returns None

Return type None

3.2 moiety_modeling.model

This module provides the *Moiety* class, the *Molecule* class, the *Relationship* class, and the *Model* class to construct moiety model.

class moiety_modeling.model.**Moiety**(*name*, *maxIsotopeNum*, *states=None*, *isotopeStates=None*, *nickname=' '*, *ranking=0*)

Moiety class describes the *Moiety* entity in the moiety model.

__init__(*name*, *maxIsotopeNum*, *states=None*, *isotopeStates=None*, *nickname=' '*, *ranking=0*)

Moiety initializer.

Parameters

- **name** (*str*) – the name of the moiety.
- **maxIsotopeNum** (*dict*) – the dictionary of the labeling isotopes and the corresponding number in the moiety. eg: {'13C': 5, '15N': 3}.
- **states** (*list*) – the list of states of the moiety.
- **isotopeStates** (*dict*) – the dictionary of the labeling isotopes and the corresponding states in the moiety. eg: {'13C': [0, 2, 5], '15N': [2, 3]}.

- **nickname** (*str*) – the nickname of the moiety.
- **ranking** (*int*) – the ranking of the moiety.

class moiety_modeling.model.**Relationship**(*moiety*, *moietyState*, *equivalentMoiety*, *equivalentMoietyState*, *operator=None*, *coefficient=None*)

Relationship class describes the relationship between moiety states in the moiety model.

__init__(*moiety*, *moietyState*, *equivalentMoiety*, *equivalentMoietyState*, *operator=None*, *coefficient=None*)

Relationship initializer.

Parameters

- **moiety** (*Moiety*) – the *Moiety* in the relationship.
- **moietyState** (*str*) – the state of the moiety (eg: '13C_0.15N_1').
- **equivalentMoiety** (*Moiety*) – *Moiety* in the relationship.
- **equivalentMoietyState** (*str*) – the state of the equivalentMoiety (eg: '13C_0.15N_1').
- **operator** (*str*) – the operator of the relationship ('*', '/').
- **coefficient** (*double*) – the coefficient of the relationship.

class moiety_modeling.model.**Molecule**(*name*, *moieties*)

Molecule class describes the *Molecule* entity in the moiety model

__init__(*name*, *moieties*)

Molecule initializer.

Parameters

- **name** (*str*) – the name of the molecule.
- **moieties** (*list*) – the list of *Moiety* instances that make up for the *Molecule*.

__eq__(*other*)

Return self==value.

class moiety_modeling.model.**Model**(*name*, *moieties*, *molecules*, *relationships=None*)

Model class describes the moiety *Model* entity.

__init__(*name*, *moieties*, *molecules*, *relationships=None*)

Model initializer.

Parameters

- **name** (*str*) – the name of the model.
- **moieties** (*list*) – a list of *Moiety* instances.
- **molecules** (*list*) – a list of *Molecule* instances.
- **relationships** (*list*) – a list of *Relationship* instances.

3.3 moiety_modeling.analysis

This module provides several classes to analyze the optimization results, select the optimal model, and visualize the results. The `ResultAnalysis` class is responsible for generating general statistics from the optimization results. The *ModelRank* class selects the model that best reflects the observed isotopologue profile. The *ComparisonTable*

class compares the optimal model selected under different optimization settings. The *PlotMoietyDistribution* class plots the distribution of moiety value of the moiety model. The *PlotIsotopologueIntensity* class plots comparison of the observed and the calculated isotopologue intensity.

class moiety_modeling.analysis.**ResultsAnalysis**(*filename*, *path=None*)
ResultsAnalysis class performs the analysis of moiety model optimization results.

__init__(*filename*, *path=None*)
ResultsAnalysis initializer.

Parameters

- **filename** (*str*) – the filenames of optimization results file.
- **path** (*str*) – the path to store the analysis results.

analyze()
Analyze the optimization results for each model.

Return dict the analysis results.

class moiety_modeling.analysis.**ModelRank**(*pathFile*, *path*, *selectionCriterion*)
ModelRank class ranks the models according to the selection criteria.

__init__(*pathFile*, *path*, *selectionCriterion*)
ModelRank initializer.

Parameters

- **pathFile** (*str*) – the txt file containing paths to the model analysis results.
- **path** (*str*) – the path to store the model rank results.
- **selectionCriterion** (*str*) – the selection criteria (eg: AIC, BIC, BICc).

rank()
To rank the models according to the selection criteria.

Return list the rank results.

class moiety_modeling.analysis.**ComparisonTable**(*pathFile*, *path*)
ComparisonTable class collects the best and second best models under different optimization settings.

__init__(*pathFile*, *path*)
ComparisonTable initializer.

Parameters

- **pathFile** (*str*) – the filenames of model rank at different situations.
- **path** (*str*) – the path to directory that stores the result.

makeTable()
To make the comparison table under different optimization settings.

Returns None

Return type None

class moiety_modeling.analysis.**PlotMoietyDistribution**(*filename*, *path*)
PlotMoietyDistribution class plots the moiety state distribution of the optimization results.

__init__(*filename*, *path*)
PlotMoietyDistribution initializer.

Parameters

- **filename** (*str*) – the json file of model analysis results.
- **path** (*str*) – the path to store the plot results.

plotMoiety()
To plot the distribution of moiety states.

Returns None

Return type None

class moiety_modeling.analysis.**PlotIsotopologueIntensity**(*filename*, *path*)
PlotIsotopologueIntensity class plots the comparison of calculated and observed isotoplogue intensity.

__init__(*filename*, *path*)
PlotIsotopologueIntensity initializer.

Parameters

- **filename** (*str*) – The json file of model analysis results.
- **path** (*str*) – the path to store the plot results.

plotIsotopologue()
To plot the comparison of observed and calculated isotoplogue intensities.

Returns None

Return type None

3.4 The moiety_modeling command-line interface option

Usage: moiety_modeling -h | –help moiety_modeling –version moiety_modeling modeling [–combinedData=<combined_jsonfile>] [–models=<models_jsonfile] [–datasets=<datasets_jsonfile>] [–optimizations=<optimizations_jsonfile>] [–working=<working_dir>] [–repetition=<optim_count>] [–split] [–force] [–multiprocess] [–energyFunction=<function>] [–printOptimizationScripts] moiety_modeling analyze optimizations –a <optimizationPaths_txtfile> [–working=<working_dir>] moiety_modeling analyze optimizations –s <optimizationResults_jsonfile> [–working=<working_dir>] moiety_modeling analyze rank <analysisPaths_txtfile> [–working=<working_dir>] [–rankCriteria=<rankCriteria>] moiety_modeling analyze table <rankPaths_txtfile> [–working=<working_dir>] moiety_modeling plot moiety <analysisResults_jsonfile> [–working=<working_dir>] moiety_modeling plot isotopologue <analysisResults_jsonfile> [–working=<working_dir>]

Options:

-h, --help Show this screen.

--version Show version.

--combinedData=<combined_jsonfile> JSON description file of the combined data (eg: models, datasets, optimization settings)

--models=<models_jsonfile> JSON description file of the moiety models.

--datasets=<datasets_jsonfile> JSON description file of the datasets.

--optimizations=<optimizations_jsonfile> JSON description file of the optimization setting.

--working=<working_dir> Alternative path to save the results.

--repetition=<optim_count> The number of optimization repetitions to perform [default: 100].

--split To split the datasets or not.

--force To force optimization process if error occurs.

--multiprocess To perform with multiprocessing or not.

--printOptimizationScripts To print the optimization script or not.

--a To analyze a bunch of optimization results together with the path file.

--s To analyze a single moiety model optimization results.

--energyFunction=<enegyFunction> The energyFunction of the moiety modeling optimization.

--optimizationSetting=<optimizationSetting> The optimization setting of the moiety modeling optimization.

--rankCriteria=<rankCriteria> The criteria for model ranking [default: AICc].

moiety_modeling.cli.**cli**(*args*)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

Symbols

A

B

C

D

E

L

M

O

P

R

S