Atomically resolve a metabolic reconstruction

Author: German Preciat, Analytical BioSciences, Leiden University

INTRODUCTION

Genome-scale metabolic network reconstructions have become a relevant tool in modern biology to study the metabolic pathways of biological systems *in silico*. However, a more detailed representation at the underlying level of atom mappings opens the possibility for a broader range of biological, biomedical and biotechnological applications than with stoichiometry alone.

This tutorial will demonstrate how to use the chimoinformatic tools in the COBRA Toolbox. The tutorial is divided into three sections: first, the chimoinformatic data of the metabolites in a COBRA model is processed generating metabolite structures in various chemoinfomatic formats; second, the atoms of their reactions are mapped; and finally, all of the tools demonstrated are used to generate a standardized chemoinformatic database specific to the COBRA model. The chemoinformatic database will be generated using information from the ecoliCore model.

MATERIALS

To atom map reactions it is required to have Java version 8 and Linux. The atom mapping does not run on Windows at present.

On *macOS*, please make sure that you run the following commands in the Terminal before continuing with this tutorial:

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

```
$ brew install coreutils
```

On *Linux*, please make sure that Java and ChemAxon directories are included. To do this, run the following commands:

`$ export PATH=$PATH:/opt/opt/chemaxon/jchemsuite/bin/` (default location of JChem)

`$ export PATH=$PATH:/usr/java/jre1.8.0_131/bin/` (default installation of Java)

Also, in order to standardise the chemical reaction format, it is required to have JChem downloaded from ChemAxon with its respective license.

Metabolites

Metabolite structures are represented in a variety of chemoinformatic formats, including 1) Metabolite chemical tables (MDL MOL) that list all of the atoms in a molecule, as well as their coordinates and bonds [1]; 2) The simplified molecular-input line-entry system (SMILES), which uses a string of ASCII characters to describe the structure of a molecule [2]; or 3) The International Chemical Identifier (InChI) developed by the IUPAC, provides a standard representation for encoding molecular structures using multiple layers to describe a metabolite structure [3] (see Figure 1). Additionally, different chemical databases assing a particular identifier to represent

the metabolite structures as the Virtual Metabolic Human database (**VMH**) [4], the Human Metabolome Database (**HMDB**) [5], **PubChem** database [6], the Kyoto Encyclopedia of Genes and Genomes(**KEEG**) [7], and the Chemical Entities of Biological Interest (**ChEBI**) [8].

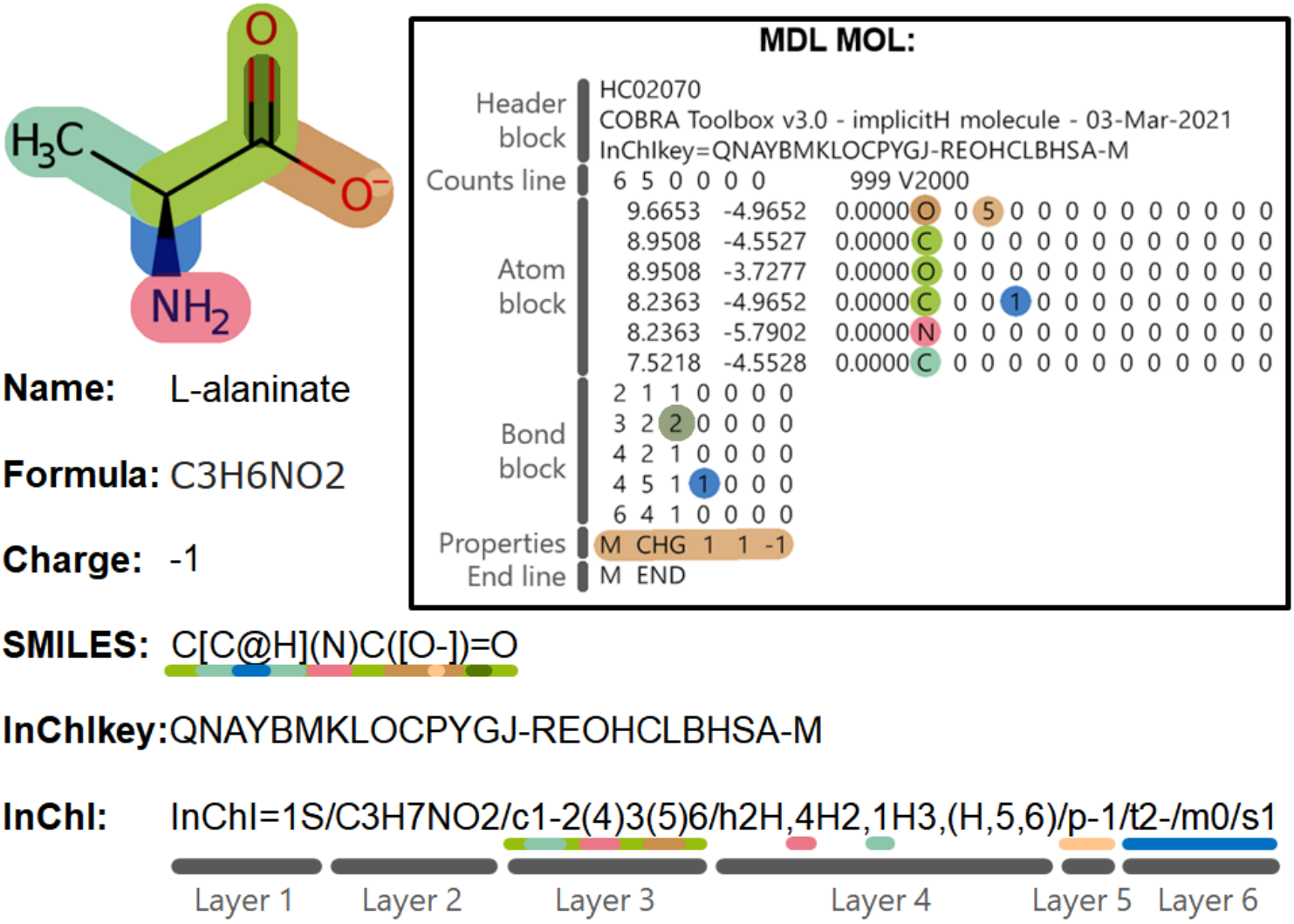


Figure 1. L-alaninate molecule represented by a hydrogen-suppressed molecular graph (implicit hydrogens). The main branch of the molecule can be seen in green; the additional branches can be seen in brown, pink and turquoise. The stereochemistry of the molecule is highlighted in blue, the double bond with dark green and the charges are highlighted in light brown. The same colours are used to indicate where this information is represented in the different chemoinformatic formats. The InChI is divided into layers, each of which begins with a lowercase letter, except for Layers 1 and 2. Layer 1 indicates if the InChI is standardised, Layer 2 the chemical formula in a neutral state, Layer 3 the connectivity between the atoms (ignoring hydrogen atoms), Layer 4 the connectivity of hydrogen atoms, Layer 5 the charge of the molecule and Layer 6 the stereochemistry. Additional layers can be added, but they cannot be represented with a standard InChI.

First we clean the workspace and load the model.

```
clear
load ecoli_core_model.mat
model.mets = regexprep(model.mets, '\-', '\_');
```

Add metabolite information

The `addMetInfoInCBmodel` function will be used to add the identifiers. The chemoinformatic data is obtained from an external file and is added to the ecoliCore model. The chemoinformatic information includes SMILES, InChIs, or different database identifiers.

```
dataFile = which('chemoinformaticDatabaseTutorial.mlx');
inputData = regexprep(dataFile, 'chemoinformaticDatabaseTutorial.mlx', 'metaboliteIds.
replace = false;
[model, hasEffect] = addMetInfoInCBmodel(model, inputData, replace);
clearvars -except model
```

Download metabolites from model identifiers

The `obtainMetStructures` function is used to obtain MDL MOL files from different databases, including HMDB [5], PubChem [6], KEEG [7] and ChEBI [8]. Alternatively, the function can be used to convert the InChI strings or SMILES in the model to MDL MOL files. A COBRA model with identifiers is required to run the function.

The optional variables are:

The variable lmetsl contains a list of metabolites to be download (Default: All). To obtain the metabolite structure of glucose, we use the VMH id.

```
mets = {'2pg'; 'h2o'; 'pep'; 'fdp'; 'f6p'; 'pi'};
```

`outputDir`: Path to the directory that will contain the MOL files (default: current directory).

```
outputDir = [pwd filesep 'comparison' filesep];
```

`sources`, is an array indicating the source of preference (default: all the sources with ID)

1. InChI (requires openBabel)
2. Smiles (requires openBabel)
3. KEGG (https://www.genome.jp/)
4. HMDB (https://hmdb.ca/)
5. PubChem (https://pubchem.ncbi.nlm.nih.gov/)
6. CHEBI (https://www.ebi.ac.uk/)

```
sources = {'inchi'; 'smiles'; 'kegg'; 'hmdb'; 'pubchem'; 'chebi'};
```

Run the function

```
molCollectionReport = obtainMetStructures(model, mets, outputDir, sources);
```

Convert metabolites

Open Babel [9] is a chemical toolbox designed to translate the different chemical data languages. It is possible to convert between chemical formats such as MDL MOL files to InChI. This function `openBabelConverter` converts chemoformatic formats using OpenBabel. It requires having OpenBabel installed.

The function requires the original chemoinformatic structure (`origFormat`) and the output format (`outputFormat`). The formats supported are SMILES, MD MOL, InChI, InChIKey, rxn and rinchi. Furthermore, if the optional variable `saveFileDir` is set, the new format will be saved with the name specified in the variable.

All of the downloaded metabolite structures are converted to an InChI as follows.

```
[inchis, smiles] = deal(cell(size(mets)));
for i = 1:length(sources)
    metaboliteDir = [outputDir 'metabolites' filesep sources{i} filesep];
    inchis{i, 1} = openBabelConverter([metaboliteDir 'f6p.mol'], 'inchi');
    smiles{i, 1} = openBabelConverter(inchis{i, 1}, 'smiles');
end
table(mets, inchis, smiles)
```

ans = 6×3 table

	mets	inchis	smiles
1	'2pg'	'InChI=1S/C...	'C(C1C(C...
2	'h2o'	'InChI=1S/C...	'C(C1C(C...
3	'pep'	'InChI=1S/C...	'C([C@@H...
4	'fdp'	'InChI=1S/C...	'C(C(=O)...
5	'f6p'	'InChI=1S/C...	'C(C(=O)...
6	'pi'	'InChI=1S/C...	'C(C(=O)...

InChI comparison

With the function `compareInchis`, each InChI string is given a score based on its similarity to the chemical formula and charge of the metabolite in the model. Factors such as stereochemistry, if it is a standard inchi, and its similarity to the other inchis are also considered. The InChI with the highest score is the identifier considered as more consistent with the model.

```
comparisonTable = compareInchis(model, inchis, 'f6p');
display(comparisonTable)
```

comparisonTable = 6×15 table

...

	scores	rGroup	InChI	metFormula	formulaOkBool	netCharge
1	12.8333	0	'InChI=1S/C...	"C6H13O9P"	1	-2
2	12.8333	0	'InChI=1S/C...	"C6H13O9P"	1	-2
3	13.6667	0	'InChI=1S/C...	"C6H13O9P"	1	0
4	14	0	'InChI=1S/C...	"C6H13O9P"	1	0
5	14	0	'InChI=1S/C...	"C6H13O9P"	1	0
6	14	0	'InChI=1S/C...	"C6H13O9P"	1	0

Metabolite structure standardisation

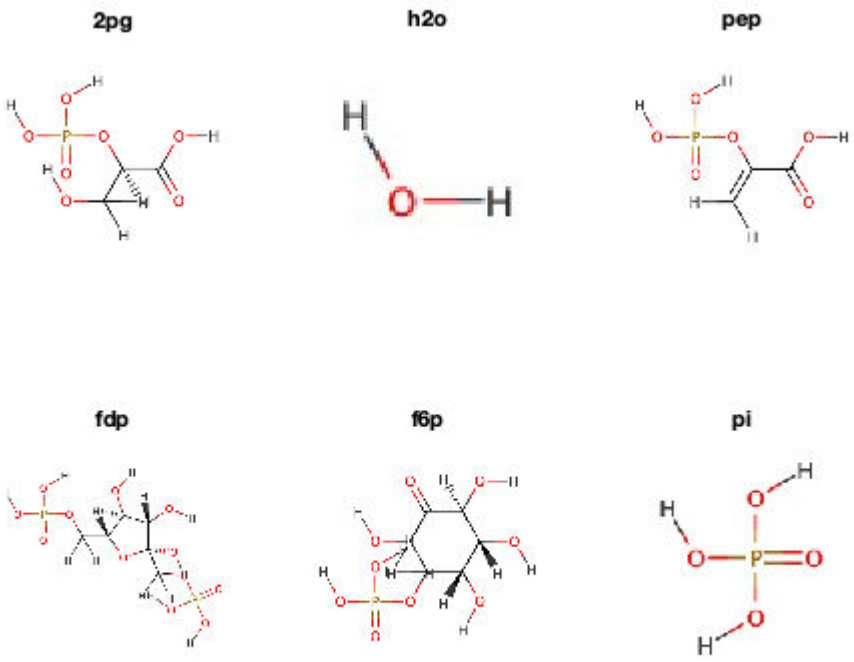
1. explicitH: Chemical graphs;
2. implicitH: Hydrogen suppressed chemical graph;
3. basic: Update the header.

```
hmdbDir = [outputDir 'metabolites' filesep 'hmdb' filesep];
metList = mets;
standardisedDir = [pwd filesep 'mets' filesep];
standardisationApproach = 'explicitH';
standardisationReport = standardiseMolDatabase(hmdbDir, metList, ...
    standardisedDir, standardisationApproach);
```

```
Standardizing 6 MOL files ...
```

Metabolite structures

```
imagesFolder = [standardisedDir 'images' filesep];
figure
for i = 1:length(metList)
    subplot(2, 3, i)
    imshow([imagesFolder metList{i} '.jpeg'])
    title(metList{i})
end
```



Reactions

A set of atom mappings represents the mechanism of each chemical reaction in a metabolic network, each of which relates an atom in a substrate metabolite to an atom of the same element in a product metabolite (Figure 1). To atom map reactions in a metabolic network reconstruction, one requires chemical structures in a data file format (SMILES, MDL MOL, InChIs), reaction stoichiometries, and an atom mapping algorithm.

A set of atom mappings represents the mechanism of each chemical reaction in a metabolic network, each of which relates an atom in a substrate metabolite to an atom of the same element in a product metabolite (Figure 1). To atom map reactions in a metabolic network reconstruction, one requires chemical structures in a data file format (SMILES, MDL MOL and InChIs), reaction stoichiometries, and an atom mapping algorithm.

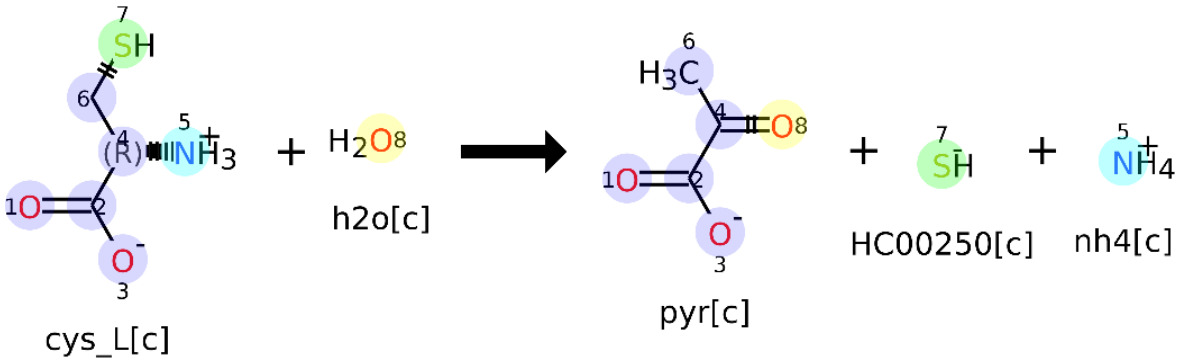


Figure 1. Set of atom mappings for reaction L-Cysteine L-Homocysteine-Lyase (VMH ID: r0193).

Metabolite structures and reaction stoichiometries from the genome-scale reconstruction are used to generate reaction chemical tables containing information about the chemical reactions (MDL RXN). The metabolic reactions are atom mapped using the Reaction Decoder Tool (RDT) algorithm [11], which was chosen after comparing the performance of published atom mapping algorithms [12]. Atom map metabolic reactions Atom mappings for the internal reactions of a metabolic network reconstruction are performed by the function `obtainAtomMappingsRDT`.

For this section, the atom mappings are generated based on the molecular structures obtained and the ecoli core model.

The function `obtainAtomMappingsRDT` generates 4 different directories containing:

- the atom mapped reactions in MDL RXN format (directory *atomMapped*),
- the images of the atom mapped reactions (directory *images*),
- additional data for the atom mapped reactions (SMILES, and product and reactant indexes) (directory *txtData*), and
- the unmapped MDL RXN files (directory *rxnFiles*).

The input variable `outputDir` indicates the directory where the folders will be generated (by default the function assigns the current directory).

Atom map a reaction

The main inputs of the `obtainAtomMappingsRDT function` are a COBRA model structure and a directory containing the molecular structures in MDL MOL format. The variable `molFileDir` contains the path to the directory containing MOL files of the COBRA model.

```
molFileDir = [standardisedDir filesep 'molFiles'];
```

The variable `rxnDir` specifies the path to the directory containing the RXN files with atom mappings.

```
rxnDir = [pwd filesep 'rxns'];
```

The input variable `rxnsToAM` indicates the reactions that will be atom mapped. By default the function atom map all the internal reactions with all of its metabolites present in the metabolite database (`molFileDir`).

```
rxnsToAM = {'ENO'; 'FBP'};
```

The variable `hMapping`, indicates if the hydrogen atoms will be also atom mapped (Default: `true`).

```
hMapping = true;
```

Finally, the variable `onlyUnmapped` indicates if only the reaction files will be generated without atom mappings (Default: `false`).

```
onlyUnmapped = false;
```

Now, let's obtain the atom map using `obtainAtomMappingsRDT`:

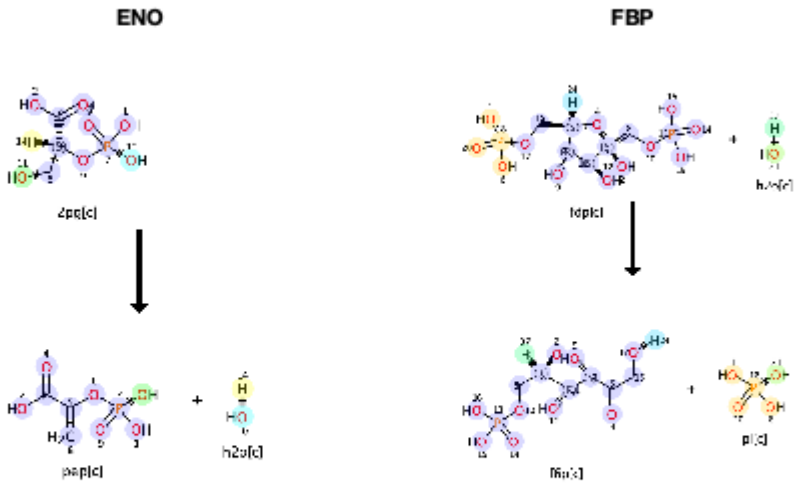
```
atomMappingReport = obtainAtomMappingsRDT(model, molFileDir, rxnDir, rxnsToAM, hMapping
```

```
Generating RXN files.
Computing atom mappings for 2 reactions.
```

The output, `atomMappingReport`, contains a report of the reactions written which include:

- `rxnFilesWritten`: The MDL RXN written.
- `balanced`: The atomically balanced reactions.
- `unbalanced`: The atomically unbalanced reactions.
- mapped: The atom mapped reactions.
- `notMapped`: The unmapped reactions.
- `inconsistentBool`: A Boolean vector indicating the inconsistent reactions.
- `rinchi`: The reaction InChI for the MDL RXN files written.
- `rsmi`: The reaction SMILES for the MDL RXN files written.

```
imagesFolder = [rxnDir filesep 'images' filesep];
figure
for i = 1:length(rxnsToAM)
    subplot(1, 2, i)
    imshow([imagesFolder rxnsToAM{i} '.png'])
    title(rxnsToAM{i})
end
```



Read atom mapping data

The information of an atom mapped reaction is extracted using the function `readAtomMappingFromRxnFile`, which includes a metabolite identifier, the element, the atom mapping index, the identification of substrate or product, and a vector indicating which instance of a repeated metabolite atom I belongs to. The atom mapped data for the reaction enolase is extracted in the following example.

```
atomMapDir = [rxnDir filesep 'atomMapped'];
[mets, elements, metNrs, rxnNrs, isSubstrate, instances] = readAtomMappingFromRxnFile(
display(table(mets, elements, metNrs, rxnNrs, isSubstrate, instances))
```

```
  36×6 table

      mets          elements     metNrs    rxnNrs    isSubstrate    instances
    _________    ____________    ______    ______    ___________    _________

    {'2pg[c]'}     {'H'}          1         1          true            1
    {'2pg[c]'}     {'O'}          2         2          true            1
    {'2pg[c]'}     {'C'}          3         3          true            1
    {'2pg[c]'}     {'O'}          4         4          true            1
    {'2pg[c]'}     {'C'}          5         5          true            1
    {'2pg[c]'}     {'H'}          6         6          true            1
    {'2pg[c]'}     {'O'}          7         7          true            1
    {'2pg[c]'}     {'P'}          8         8          true            1
    {'2pg[c]'}     {'O'}          9         9          true            1
    {'2pg[c]'}     {'O'}         10        10          true            1
    {'2pg[c]'}     {'H'}         11        11          true            1
    {'2pg[c]'}     {'O'}         12        12          true            1
    {'2pg[c]'}     {'H'}         13        13          true            1
    {'2pg[c]'}     {'C'}         14        14          true            1
    {'2pg[c]'}     {'H'}         15        15          true            1
```

```
{'2pg[c]'}    {'H'}      16      16      true          1
{'2pg[c]'}    {'O'}      17      17      true          1
{'2pg[c]'}    {'H'}      18      18      true          1
{'h2o[c]'}    {'H'}       1       6      false         1
{'h2o[c]'}    {'O'}       2      12      false         1
{'h2o[c]'}    {'H'}       3      13      false         1
{'pep[c]'}    {'H'}       1       1      false         1
{'pep[c]'}    {'O'}       2       2      false         1
{'pep[c]'}    {'C'}       3       3      false         1
{'pep[c]'}    {'O'}       4       4      false         1
{'pep[c]'}    {'C'}       5       5      false         1
{'pep[c]'}    {'O'}       6       7      false         1
{'pep[c]'}    {'P'}       7       8      false         1
{'pep[c]'}    {'O'}       8       9      false         1
{'pep[c]'}    {'O'}       9      10      false         1
{'pep[c]'}    {'H'}      10      11      false         1
{'pep[c]'}    {'O'}      11      17      false         1
{'pep[c]'}    {'H'}      12      18      false         1
{'pep[c]'}    {'C'}      13      14      false         1
{'pep[c]'}    {'H'}      14      16      false         1
{'pep[c]'}    {'H'}      15      15      false         1
```

Find the enthalpy change and number of bonds broken and formed

The `findEnthalpyChange` and `findBondsBrokenAndFormed` functions are used to calculate the enthalpy change or the number of broken and formed bonds of each reaction in list `rxnsToAM` using the reaction mechanism identified by the atom mapping. Furthermore, the total weight of all substrates is calculated by adding the atomic weight of each atom.

```
[enthalpyChange, substrateMass] = findEnthalpyChange(model, rxnsToAM, atomMapDir);
[bondsBrokenAndFormed, ~] = findBondsBrokenAndFormed(model, rxnsToAM, atomMapDir);
```

Make a table of enthalpy change, bonds broken and formed, and sort it by modified bonds.

```
rxnDataTable = table(rxnsToAM, bondsBrokenAndFormed, enthalpyChange, substrateMass, ..
    'VariableNames', {'rxns','bondsBrokenAndFormed', 'enthalpyChange', 'substrateMass'
rxnDataTable = sortrows(rxnDataTable, {'bondsBrokenAndFormed'}, {'descend'});
display(rxnDataTable)
```

`rxnDataTable = 2×4 table`

	rxns	bondsBrokenAndFormed	enthalpyChange	substrateMass
1	'FBP'	8	-83	358.1332
2	'ENO'	6	54	186.0583

Chemoinformatic database

The function `generateChemicalDatabase` generates a chemoinformatic database of standardised metabolite structures and atom-mapped reactions on a genome-scale metabolic reconstruction using the tools described in this tutorial. In order to identify the metabolite structure that most closely resembles the metabolite in the genome-scale reconstruction, identifiers from different sources are compared based on their InChI (See Table 1). Finally, the obtained atom mapped reactions are used to identify the number of broken and formed bonds, as well as the enthalpy change of the reactions in the genome-scale reconstruction.

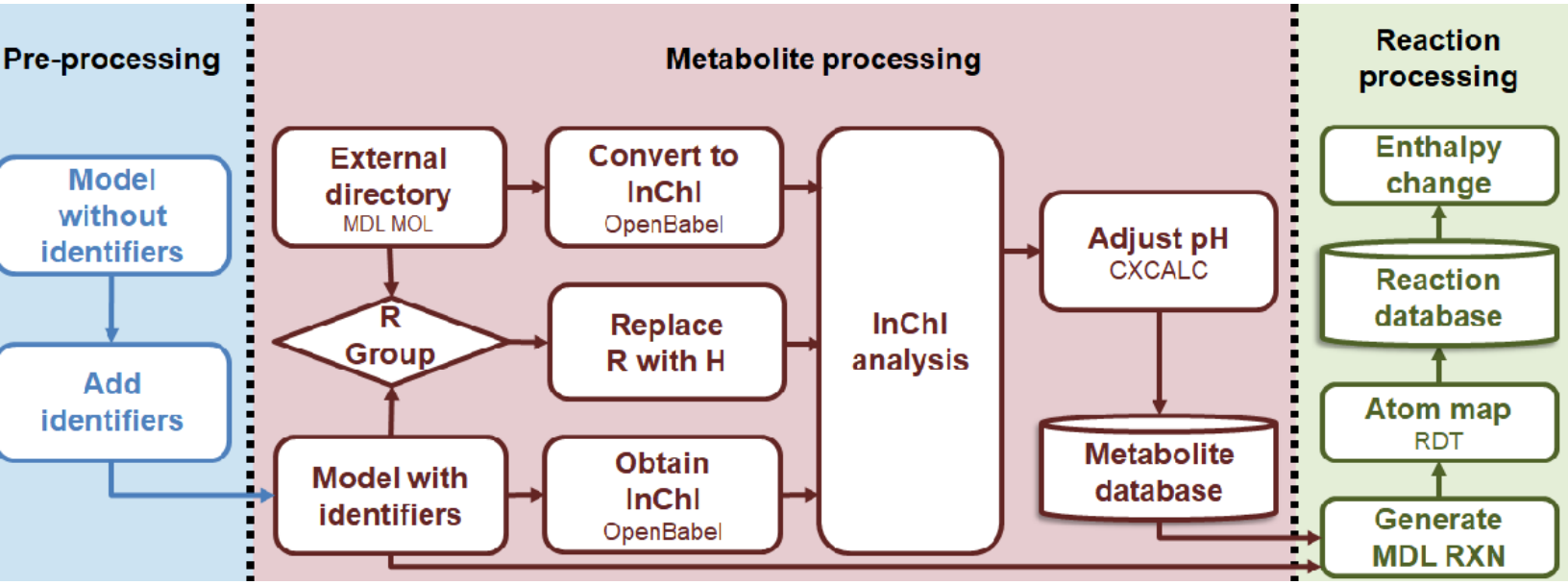


Figure 2. `generateChemicalDatabase` workflow

Table 1. InChI scoring criteria.

Concept	Score	Description
Chemical formula	0 or 10	The chemical formula indicated in the genome-scale model is compared with that obtained from the InChI. This feature is given more weight in order to keep the metabolite as described in the genome-scale model. Hydrogen atoms are ignored in this comparison since they can be modified based on the charge (Figure 1).
Charge	0 or 1	The charge indicated in the genome-scale model is compared with the charge obtained from the source.
Stereochemical information	0 or 1	Indicates whether the InChI contains stereochemical information or not.
Standard	0 or 1	Indicates whether the InChI is standardised or not.
Similarity with other databases	0-1	The number of sources where the InChI strings are identical, divided by the total number of sources.
Main layer similarity	0-1	The number of sources where the main layers are identical, divided by the total number of sources.
InChI with more layers	0 or 1	InChI with more layers.

The goal of the comparison is to obtain a larger number of atomically balanced metabolic reactions. The Reaction Decoder Tool algorithm [8] (**RDT**) is used to obtain the atom mappings of each metabolic reaction. The atom mapping data is used to calculate the number of bonds formed or broken in a metabolic reaction, as well as the enthalpy change. The information gathered is incorporated into the COBRA model.

We will obtain chemoinformatic database of the Ecoli core model in this tutorial.

The user-defined parameters in the function `generateChemicalDatabase` will activate various processes. Each parameter is contained in the struct array `options` and described in detail below:

- **outputDir**: The path to the directory containing the chemoinformatic database (default: current directory)
- **printlevel**: Verbose level
- **standardisationApproach**: String containing the type of standardisation for the molecules (default: 'explicitH' if openBabel [9] is installed, otherwise default: 'basic'):

1. explicitH: Chemical graphs;
2. implicitH: Hydrogen suppressed chemical graph;
3. basic: Update the header.

- **keepMolComparison**: Logical value, indicate if all metabolite structures per source will be saved or not.
- **onlyUnmapped**: Logic value to select create only unmapped MDL RXN files (default: FALSE, requires Java to run the RDT [11]).
- **adjustToModelpH**: Logic value used to determine whether a molecule's pH must be adjusted in accordance with the COBRA model. (default: TRUE, requires MarvinSuite [10]).
- **addDirsToCompare**: Cell(s) with the path to directory to an existing database (default: empty).
- **dirNames**: Cell(s) with the name of the directory(ies) (default: empty).
- **debug**: Logical value used to determine whether or not the results of different points in the function will be saved for debugging (default: empty).

```
options.outputDir = [pwd filesep 'database'];
options.printlevel = 1;
options.debug = true;
options.standardisationApproach = 'explicitH';
options.adjustToModelpH = true;
options.keepMolComparison = false;
options.dirsToCompare = {['~' filesep 'work' filesep 'code' filesep 'ctf' filesep 'mets
options.onlyUnmapped = false;
options.dirNames = {'VMH'};
```

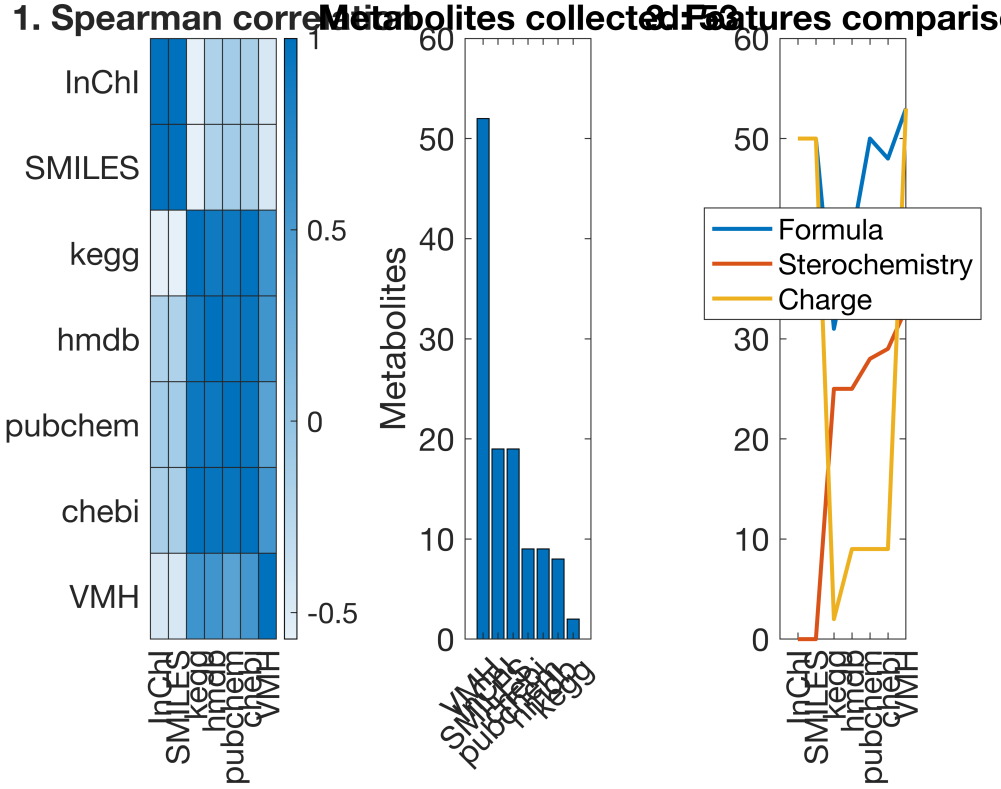
Use the function `generateChemicalDatabase`

```
info = generateChemicalDatabase(model, options);
```

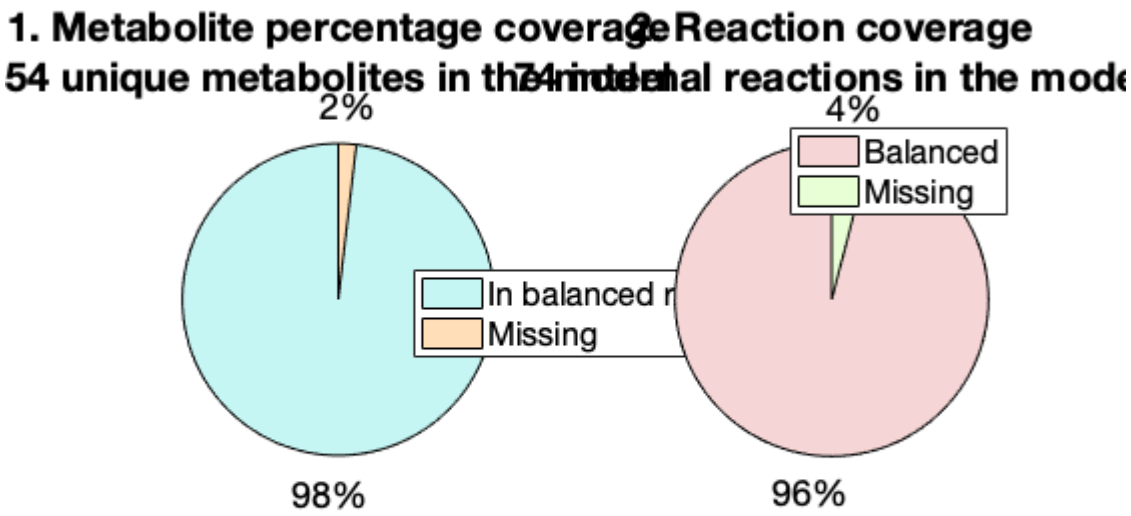
```
------------------------------------------------------------
CHEMICAL DATABASE
------------------------------------------------------------

Generating a chemical database with the following options:

                  outputDir: '/Users/gpreciat/Desktop/asd/database'
                 printlevel: 1
                      debug: 1
    standardisationApproach: 'explicitH'
            adjustToModelpH: 1
          keepMolComparison: 0
              dirsToCompare: {'~/work/code/ctf/mets/molFiles/'}
               onlyUnmapped: 0
                   dirNames: {'VMH'}

------------------------------------------------------------
Obtaining MOL files from chemical databases ...

inchi:
molCollectionReport = struct with fields:
               mets: {54×1 cell}
       metsWithMol: {50×1 cell}
    metsWithoutMol: {4×1 cell}
```

```
          coverage: 92.5926
        idsToCheck: {}
smiles:
molCollectionReport = struct with fields:
              mets: {54×1 cell}
       metsWithMol: {50×1 cell}
    metsWithoutMol: {4×1 cell}
          coverage: 92.5926
        idsToCheck: {}
kegg:
molCollectionReport = struct with fields:
              mets: {54×1 cell}
       metsWithMol: {31×1 cell}
    metsWithoutMol: {23×1 cell}
          coverage: 57.4074
        idsToCheck: {}
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
The server returned the status 503 with message "Service Temporarily Unavailable" in response to the reque
hmdb:
molCollectionReport = struct with fields:
              mets: {54×1 cell}
       metsWithMol: {40×1 cell}
    metsWithoutMol: {14×1 cell}
          coverage: 74.0741
        idsToCheck: {11×1 cell}
pubchem:
molCollectionReport = struct with fields:
              mets: {54×1 cell}
       metsWithMol: {50×1 cell}
    metsWithoutMol: {4×1 cell}
          coverage: 92.5926
        idsToCheck: {}
chebi:
molCollectionReport = struct with fields:
              mets: {54×1 cell}
       metsWithMol: {48×1 cell}
    metsWithoutMol: {6×1 cell}
          coverage: 88.8889
        idsToCheck: {}
VMH:
  struct with fields:

              mets: {54×1 cell}
       metsWithMol: {53×1 cell}
    metsWithoutMol: {'acon_C'}
          coverage: 98.1481

Comparing information from sources ...
  53×6 table

       mets                      source                         score
    __________    _______________________________________    ______    ___________________________________

    {'13dpg' }    {'VMH'                                 }    15.167    {'InChI=1S/C3H8O10P2/c4-2(1-12-14(
    {'2pg'   }    {'VMH'                                 }    15.143    {'InChI=1S/C3H7O7P/c4-1-2(3(5)6)10-
```

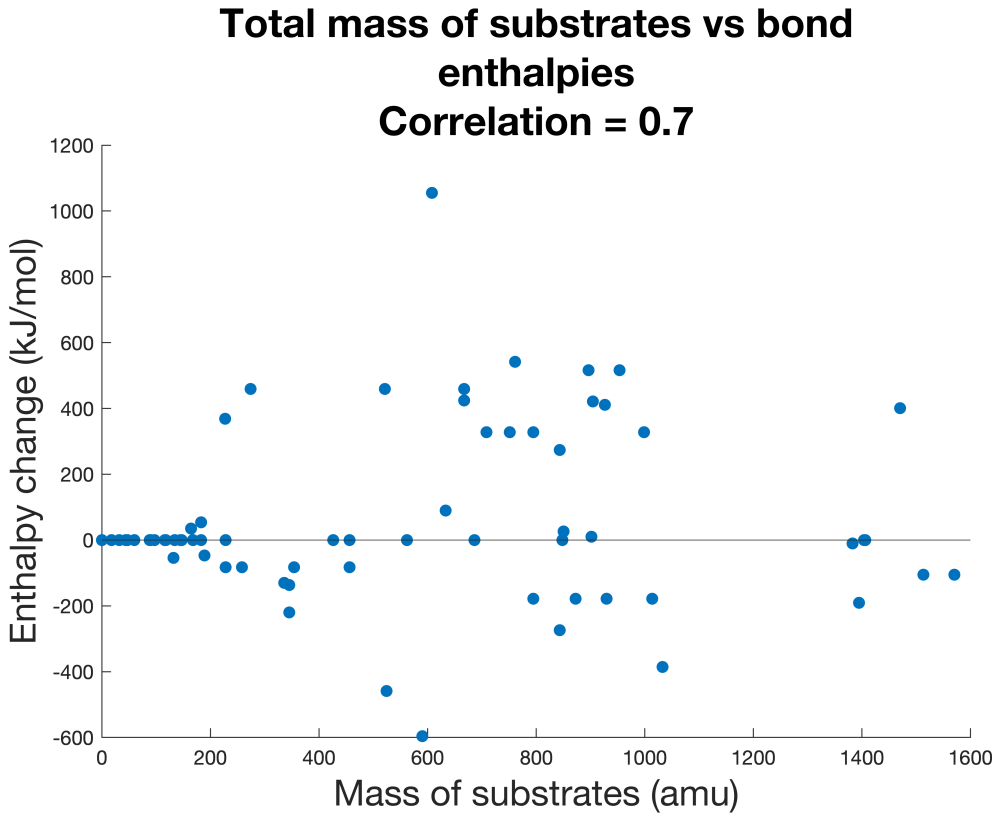
{'3pg' }	{'VMH' }	15.143	{'InChI=1S/C3H7O7P/c4-2(3(5)6)1-10-
{'6pgc' }	{'VMH' }	15.143	{'InChI=1S/C6H13O10P/c7-2(1-16-17(1
{'6pgl' }	{'VMH' }	15.143	{'InChI=1S/C6H11O9P/c7-3-2(1-14-16(
{'ac' }	{'inchi smiles VMH' }	14.5	{'InChI=1S/C2H4O2/c1-2(3)4/h1H3,(H,
{'acald' }	{'inchi smiles hmdb pubchem chebi VMH'}	15	{'InChI=1S/C2H4O/c1-2-3/h2H,1H3'
{'accoa' }	{'VMH' }	15.143	{'InChI=1S/C23H38N7O17P3S/c1-12(31)
{'adp' }	{'VMH' }	15.143	{'InChI=1S/C10H15N5O10P2/c11-8-5-9(
{'akg' }	{'inchi smiles VMH' }	14.5	{'InChI=1S/C5H6O5/c6-3(5(9)10)1-2-4
{'amp' }	{'VMH' }	15.143	{'InChI=1S/C10H14N5O7P/c11-8-5-9(13
{'atp' }	{'VMH' }	15.143	{'InChI=1S/C10H16N5O13P3/c11-8-5-9(
{'cit' }	{'inchi smiles VMH' }	14.429	{'InChI=1S/C6H8O7/c7-3(8)1-6(13,5(1
{'co2' }	{'inchi smiles hmdb pubchem chebi VMH'}	15	{'InChI=1S/CO2/c2-1-3'
{'coa' }	{'VMH' }	15.167	{'InChI=1S/C21H36N7O16P3S/c1-21(2,1
{'dhap' }	{'inchi smiles VMH' }	14.429	{'InChI=1S/C3H7O6P/c4-1-3(5)2-9-10(
{'e4p' }	{'VMH' }	15.143	{'InChI=1S/C4H9O7P/c5-1-3(6)4(7)2-1
{'etoh' }	{'inchi smiles hmdb pubchem chebi VMH'}	15	{'InChI=1S/C2H6O/c1-2-3/h3H,2H2,1H3
{'f6p' }	{'VMH' }	14.714	{'InChI=1S/C6H13O9P/c7-1-3(8)5(10)6
{'fdp' }	{'VMH' }	15	{'InChI=1S/C6H14O12P2/c7-4-3(1-16-1
{'for' }	{'inchi smiles VMH' }	14.5	{'InChI=1S/CH2O2/c2-1-3/h1H,(H,2,3)
{'fru' }	{'kegg hmdb pubchem chebi VMH' }	15.714	{'InChI=1S/C6H12O6/c7-1-3-4(9)5(10)
{'fum' }	{'VMH' }	15.167	{'InChI=1S/C4H4O4/c5-3(6)1-2-4(7)8/
{'g3p' }	{'VMH' }	15.167	{'InChI=1S/C3H7O6P/c4-1-3(5)2-9-10(
{'g6p' }	{'VMH' }	15.143	{'InChI=1S/C6H13O9P/c7-3-2(1-14-16(
{'glc_D' }	{'kegg pubchem chebi VMH' }	15.571	{'InChI=1S/C6H12O6/c7-1-2-3(8)4(9)5
{'gln_L' }	{'hmdb pubchem chebi VMH' }	15.667	{'InChI=1S/C5H10N2O3/c6-3(5(9)10)1-
{'glu_L' }	{'VMH' }	15.167	{'InChI=1S/C5H9NO4/c6-3(5(9)10)1-2-
{'glx' }	{'inchi smiles VMH' }	14.429	{'InChI=1S/C2H2O3/c3-1-2(4)5/h1H,(H
{'h' }	{'inchi smiles hmdb pubchem chebi' }	13.667	{'InChI=1S/p+1'
{'h2o' }	{'inchi smiles hmdb pubchem chebi VMH'}	15	{'InChI=1S/H2O/h1H2'
{'icit' }	{'inchi smiles VMH' }	14.429	{'InChI=1S/C6H8O7/c7-3(8)1-2(5(10)1
{'lac_D' }	{'VMH' }	15.167	{'InChI=1S/C3H6O3/c1-2(4)3(5)6/h2,4
{'mal_L' }	{'VMH' }	15.167	{'InChI=1S/C4H6O5/c5-2(4(8)9)1-3(6)
{'nad' }	{'VMH' }	15.143	{'InChI=1S/C21H27N7O14P2/c22-17-12-
{'nadh' }	{'VMH' }	15.143	{'InChI=1S/C21H29N7O14P2/c22-17-12-
{'nadp' }	{'VMH' }	15.167	{'InChI=1S/C21H28N7O17P3/c22-17-12-
{'nadph' }	{'VMH' }	15.143	{'InChI=1S/C21H30N7O17P3/c22-17-12-
{'nh4' }	{'inchi smiles VMH' }	14.2	{'InChI=1S/H3N/h1H3/p+1'
{'o2' }	{'inchi smiles pubchem chebi VMH' }	15	{'InChI=1S/O2/c1-2'
{'oaa' }	{'inchi smiles VMH' }	14.429	{'InChI=1S/C4H4O5/c5-2(4(8)9)1-3(6)
{'pep' }	{'inchi smiles VMH' }	14.5	{'InChI=1S/C3H5O6P/c1-2(3(4)5)9-10(
{'pi' }	{'inchi smiles VMH' }	14.6	{'InChI=1S/H3O4P/c1-5(2,3)4/h(H3,1,
{'pyr' }	{'inchi smiles VMH' }	14.6	{'InChI=1S/C3H4O3/c1-2(4)3(5)6/h1H3
{'r5p' }	{'VMH' }	15.167	{'InChI=1S/C5H11O8P/c6-3-2(1-12-14(
{'ru5p_D'}	{'VMH' }	15.167	{'InChI=1S/C5H11O8P/c6-1-3(7)5(9)4(
{'s7p' }	{'VMH' }	15	{'InChI=1S/C7H15O10P/c8-1-3(9)5(11)
{'succ' }	{'inchi smiles VMH' }	14.5	{'InChI=1S/C4H6O4/c5-3(6)1-2-4(7)8/
{'succoa'}	{'VMH' }	15.167	{'InChI=1S/C25H40N7O19P3S/c1-25(2,2
{'xu5p_D'}	{'VMH' }	15.167	{'InChI=1S/C5H11O8P/c6-1-3(7)5(9)4(
{'actp' }	{'VMH' }	14.5	{'InChI=1S/C2H5O5P/c1-2(3)7-8(4,5)6
{'q8h2' }	{'hmdb VMH' }	16	{'InChI=1S/C49H76O4/c1-36(2)20-13-2
{'q8' }	{'VMH' }	16	{'InChI=1S/C49H74O4/c1-36(2)20-13-2



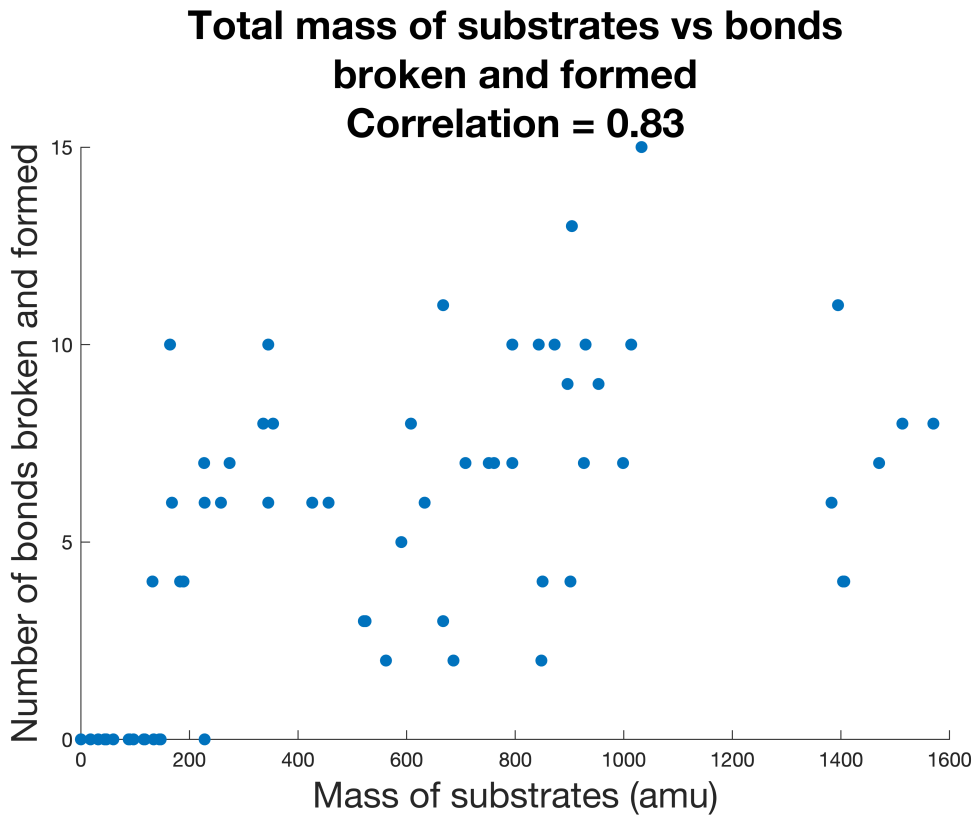

```
Adjusting pH based on the model's chemical formula ...

adjustedpH:
  53×11 table

      mets                    source                  score
```




```
Calculating bonds broken and formed and enthalpy change...
Found biomass reaction: Biomass_Ecoli_core_N(w/GAM)_Nmet2
ATP maintenance reaction is not considered an exchange reaction by default. It should be mass balanced:
ATPM    atp[c] + h2o[c]     ->    adp[c] + h[c] + pi[c]
```



```
Found biomass reaction: Biomass_Ecoli_core_N(w/GAM)-Nmet2
ATP maintenance reaction is not considered an exchange reaction by default. It should be mass balanced:
ATPM    atp[c] + h2o[c]     ->    adp[c] + h[c] + pi[c]
```



95×5 table

rxns	rxnNames
{'GLUSy' }	{'glutamate synthase (NADPH)'
{'GLUDy' }	{'glutamate dehydrogenase (NADP)'
{'GLNS' }	{'glutamine synthetase'
{'NADH16' }	{'NADH dehydrogenase (ubiquinone-8 & 3 protons)'
{'FRD7' }	{'fumarate reductase'
{'FRUpts2' }	{'Fructose transport via PEP:Pyr PTS (f6p generating)'
{'GLUN' }	{'glutaminase'
{'GND' }	{'phosphogluconate dehydrogenase'
{'ICDHyr' }	{'isocitrate dehydrogenase (NADP)'
{'ME1' }	{'malic enzyme (NAD)'
{'ME2' }	{'malic enzyme (NADP)'
{'SUCDi' }	{'succinate dehydrogenase (irreversible)'
{'CS' }	{'citrate synthase'
{'MALS' }	{'malate synthase'
{'AKGDH' }	{' 2-Oxogluterate dehydrogenase'
{'FBA' }	{'fructose-bisphosphate aldolase'
{'FBP' }	{'fructose-bisphosphatase'
{'PDH' }	{'pyruvate dehydrogenase'
{'PPS' }	{'phosphoenolpyruvate synthase'
{'ACALD' }	{'acetaldehyde dehydrogenase (acetylating)'
{'ALCD2x' }	{'alcohol dehydrogenase (ethanol)'
{'G6PDH2r' }	{'glucose 6-phosphate dehydrogenase'
{'GAPD' }	{'glyceraldehyde-3-phosphate dehydrogenase'
{'LDH_D' }	{'D-lactate dehydrogenase'
{'MDH' }	{'malate dehydrogenase'
{'PFK' }	{'phosphofructokinase'

{'PGL'	{' 6-phosphogluconolactonase'
{'PPC'	{'phosphoenolpyruvate carboxylase'
{'GLCpts'	{'D-glucose transport via PEP:Pyr PTS'
{'PGI'	{'glucose-6-phosphate isomerase'
{'PPCK'	{'phosphoenolpyruvate carboxykinase'
{'RPI'	{'ribose-5-phosphate isomerase'
{'SUCOAS'	{'succinyl-CoA synthetase (ADP-forming)'
{'TALA'	{'transaldolase'
{'TKT1'	{'transketolase'
{'TKT2'	{'transketolase'
{'TPI'	{'triose-phosphate isomerase'
{'PYK'	{'pyruvate kinase'
{'ENO'	{'enolase'
{'FUM'	{'fumarase'
{'ICL'	{'Isocitrate lyase'
{'NADTRHD'	{'NAD transhydrogenase'
{'PFL'	{'pyruvate formate lyase'
{'PGM'	{'phosphoglycerate mutase'
{'PTAr'	{'phosphotransacetylase'
{'THD2'	{'NAD(P) transhydrogenase'
{'ATPM'	{'ATP maintenance requirement'
{'ATPS4r'	{'ATP synthase (four protons for one ATP)'
{'GLNabc'	{'L-glutamine transport via ABC system'
{'ACKr'	{'acetate kinase'
{'ADK1'	{'adenylate kinase'
{'PGK'	{'phosphoglycerate kinase'
{'ACALDt'	{'acetaldehyde reversible transport'
{'ACt2r'	{'acetate reversible transport via proton symport'
{'AKGt2r'	{' 2-oxoglutarate reversible transport via symport'
{'Biomass_Ecoli_core_N(w/GAM)-Nmet2'}	{'core E. coli biomass equation (Neidhardt Based with GAM, N
{'CO2t'	{'CO2 transporter via diffusion'
{'D-LACt2'	{'D-lactate transport via proton symport'
{'ETOHt2r'	{'ethanol reversible transport via proton symport'
{'EX_ac(e)'	{'Acetate exchange'
{'EX_acald(e)'	{'Acetaldehyde exchange'
{'EX_akg(e)'	{' 2-Oxoglutarate exchange'
{'EX_co2(e)'	{'CO2 exchange'
{'EX_etoh(e)'	{'Ethanol exchange'
{'EX_for(e)'	{'Formate exchange'
{'EX_fru(e)'	{'D-Fructose exchange'
{'EX_fum(e)'	{'Fumarate exchange'
{'EX_glc(e)'	{'D-Glucose exchange'
{'EX_gln-L(e)'	{'L-Glutamine exchange'
{'EX_glu-L(e)'	{'L-Glutamate exchange'
{'EX_h2o(e)'	{'H2O exchange'
{'EX_h(e)'	{'H+ exchange'
{'EX_lac-D(e)'	{'D-lactate exchange'
{'EX_mal-L(e)'	{'L-Malate exchange'
{'EX_nh4(e)'	{'Ammonia exchange'
{'EX_o2(e)'	{'O2 exchange'
{'EX_pi(e)'	{'Phosphate exchange'
{'EX_pyr(e)'	{'Pyruvate exchange'
{'EX_succ(e)'	{'Succinate exchange'
{'FORt2'	{'formate transport in via proton symport'
{'FORti'	{'formate transport via diffusion'
{'FUMt2_2'	{'Fumarate transport via proton symport (2 H)'
{'GLUt2r'	{'L-glutamate transport via proton symport, reversible'
{'H2Ot'	{'H2O transport via diffusion'
{'MALt2_2'	{'Malate transport via proton symport (2 H)'
{'NH4t'	{'ammonia reversible transport'
{'O2t'	{'o2 transport (diffusion)'
{'PIt2r'	{'phosphate reversible transport via symport'
{'PYRt2r'	{'pyruvate reversible transport via proton symport'
{'RPE'	{'ribulose 5-phosphate 3-epimerase'

```
    {'SUCCt2_2'                         }    {'succinate transport via proton symport (2 H)'
    {'SUCCt3'                           }    {'succinate transport out via proton antiport'
    {'ACONTa'                           }    {'aconitase (half-reaction A, Citrate hydro-lyase)'
    {'ACONTb'                           }    {'aconitase (half-reaction B, Isocitrate hydro-lyase)'
    {'CYTBD'                            }    {'cytochrome oxidase bd (ubiquinol-8: 2 protons)'

Diary written to: /Users/gpreciat/Desktop/asd/database
generateChemicalDatabase run is complete.
```

Finally, the function `metDatabaseStatus` is used to check the consistency of the metabolites in a database in relation to a COBRA model, as well as, showing the type of identifiers in the model.

```
metDir = [options.outputDir filesep 'mets' filesep 'molFiles'];
[summary, status] = metDatabaseStatus(model, metDir)
```

```
summary = struct with fields:
                            mets: 54
                      consistent: 53
             inconsistentFormula: 0
              inconsistentCharge: 0
    inconsistentChargeAndFormula: 0
                         missing: 1
                        inchiIds: 50
                       smilesIds: 50
                        chebiIds: 49
                         hmdbIds: 51
                         keggIds: 50
                      pubchemIds: 50
status = 54×13 table
```

...

	met	status	noOfIds	modelFormula	structureFormula	modelCharge
1	"13dpg"	"consistent"	5	"C3H4O10P2"	"C3H4O10P2"	-4
2	"2pg"	"consistent"	6	"C3H4O7P"	"C3H4O7P"	-3
3	"3pg"	"consistent"	6	"C3H4O7P"	"C3H4O7P"	-3
4	"6pgc"	"consistent"	6	"C6H10O10P"	"C6H10O10P"	-3
5	"6pgl"	"consistent"	6	"C6H9O9P"	"C6H9O9P"	-2
6	"ac"	"consistent"	6	"C2H3O2"	"C2H3O2"	-1
7	"acald"	"consistent"	6	"C2H4O"	"C2H4O"	0
8	"accoa"	"consistent"	6	"C23H34N7O17P3S"	"C23H34N7O17P3S"	-4
9	"acon_C"	"missing"	0	"C6H3O6"	"missing"	-3
10	"actp"	"consistent"	1	"C2H3O5P"	"C2H3O5P"	-2
11	"adp"	"consistent"	6	"C10H12N5O10P2"	"C10H12N5O10P2"	-3
12	"akg"	"consistent"	6	"C5H4O5"	"C5H4O5"	-2
13	"amp"	"consistent"	6	"C10H12N5O7P"	"C10H12N5O7P"	-2
14	"atp"	"consistent"	6	"C10H12N5O13P3"	"C10H12N5O13P3"	-4
15	"cit"	"consistent"	6	"C6H5O7"	"C6H5O7"	-3
16	"co2"	"consistent"	6	"CO2"	"CO2"	0

	met	status	noOfIds	modelFormula	structureFormula	modelCharge
17	"coa"	"consistent"	6	"C21H32N7O16P3S"	"C21H32N7O16P3S"	-4
18	"dhap"	"consistent"	6	"C3H5O6P"	"C3H5O6P"	-2
19	"e4p"	"consistent"	6	"C4H7O7P"	"C4H7O7P"	-2
20	"etoh"	"consistent"	6	"C2H6O"	"C2H6O"	0
21	"f6p"	"consistent"	6	"C6H11O9P"	"C6H11O9P"	-2
22	"fdp"	"consistent"	6	"C6H10O12P2"	"C6H10O12P2"	-4
23	"for"	"consistent"	6	"CHO2"	"CHO2"	-1
24	"fru"	"consistent"	6	"C6H12O6"	"C6H12O6"	0
25	"fum"	"consistent"	6	"C4H2O4"	"C4H2O4"	-2
26	"g3p"	"consistent"	5	"C3H5O6P"	"C3H5O6P"	-2
27	"g6p"	"consistent"	6	"C6H11O9P"	"C6H11O9P"	-2
28	"glc_D"	"consistent"	6	"C6H12O6"	"C6H12O6"	0
29	"gln_L"	"consistent"	6	"C5H10N2O3"	"C5H10N2O3"	0
30	"glu_L"	"consistent"	6	"C5H8NO4"	"C5H8NO4"	-1
31	"glx"	"consistent"	6	"C2HO3"	"C2HO3"	-1
32	"h"	"consistent"	6	"H"	"H"	1
33	"h2o"	"consistent"	6	"H2O"	"H2O"	0
34	"icit"	"consistent"	6	"C6H5O7"	"C6H5O7"	-3
35	"lac_D"	"consistent"	6	"C3H5O3"	"C3H5O3"	-1
36	"mal_L"	"consistent"	6	"C4H4O5"	"C4H4O5"	-2
37	"nad"	"consistent"	6	"C21H26N7O14P2"	"C21H26N7O14P2"	-1
38	"nadh"	"consistent"	6	"C21H27N7O14P2"	"C21H27N7O14P2"	-2
39	"nadp"	"consistent"	6	"C21H25N7O17P3"	"C21H25N7O17P3"	-3
40	"nadph"	"consistent"	6	"C21H26N7O17P3"	"C21H26N7O17P3"	-4
41	"nh4"	"consistent"	6	"H4N"	"H4N"	1
42	"o2"	"consistent"	6	"O2"	"O2"	0
43	"oaa"	"consistent"	6	"C4H2O5"	"C4H2O5"	-2
44	"pep"	"consistent"	6	"C3H2O6P"	"C3H2O6P"	-3
45	"pi"	"consistent"	6	"HO4P"	"HO4P"	-2
46	"pyr"	"consistent"	6	"C3H3O3"	"C3H3O3"	-1
47	"q8"	"consistent"	0	"C49H74O4"	"C49H74O4"	0
48	"q8h2"	"consistent"	1	"C49H76O4"	"C49H76O4"	0
49	"r5p"	"consistent"	6	"C5H9O8P"	"C5H9O8P"	-2

	met	status	noOfIds	modelFormula	structureFormula	modelCharge
50	"ru5p_D"	"consistent"	6	"C5H9O8P"	"C5H9O8P"	-2
51	"s7p"	"consistent"	6	"C7H13O10P"	"C7H13O10P"	-2
52	"succ"	"consistent"	6	"C4H4O4"	"C4H4O4"	-2
53	"succoa"	"consistent"	6	"C25H35N7O19P3S"	"C25H35N7O19P3S"	-5
54	"xu5p_D"	"consistent"	6	"C5H9O8P"	"C5H9O8P"	-2

Bibliography

1. Dalby et al., "Description of several chemical structure file formats used by computer programs developed at molecular design limited", **(2002).**
2. Anderson et al., "Smiles: A line notation and computerized interpreter for chemical structures", *Environmental research Brief* **(1987)**.
3. Helle et al., "Inchi, the iupac international chemical identifier", *Journal of Cheminformatics* **(2015)**.
4. Noronha et al., "The Virtual Metabolic Human database: integrating human and gut microbiome metabolism with nutrition and disease", *Nucleic acids research* **(2018).**
5. Wishart et al., "HMDB 4.0 — The Human Metabolome Database for 2018_"._ *Nucleic acids research* **(2018).**
6. Sunghwan et al. “PubChem in 2021: new data content and improved web interfaces.” *Nucleic acids research* **(2021).**
7. Kanehisa, and Goto. "KEGG: Kyoto Encyclopedia of Genes and Genomes". *Nucleic acids research* **(2000).**
8. Hastings et al,. "ChEBI in 2016: Improved services and an expanding collection of metabolites". *Nucleic acids research* **(2016).**
9. O'Boyle et al,. "Open Babel: An open chemical toolbox." *Journal of Cheminformatics* **(2011).**
10. "Marvin was used for drawing, displaying and characterizing chemical structures, substructures and reactions, ChemAxon (<http://www.chemaxon.com>)"
11. Rahman et al,. "Reaction Decoder Tool (RDT): Extracting Features from Chemical Reactions", Bioinformatics **(2016).**
12. Preciat et al., "Comparative evaluation of atom mapping algorithms for balanced metabolic reactions: application to Recon3d", *Journal of Cheminformatics* **(2017)**.