Input and output of reconstructions and mdoels

Author(s): Thomas Pfau, University of Luxembourg

Reviewers: Catherine Clancy

INTRODUCTION

This tutorial aims at providing information on how to load models into The COBRA Toolbox and export them to other formats.

Available input formats

The COBRA Toolbox supports the use of models in multiple formats, including:

- MAT-file format
- Systems Biology Markup Language (SBML) format
- SimPheny format
- Excel format

The most commonly used model format is a MAT-file (`.mat`) format where by a simple MATLAB `struct` contains one or more of the fields defined in The COBRA Toolbox Documentation.

MAT-file format

A model in a MAT-file format is required to follow the rules defined in The COBRA Toolbox Documentation.

SBML format

The COBRA Toolbox currently supports models formats of SBML Level 3 version 1 (as defined here) and has legacy support for older versions of SBML. It also supports the Level 3 Flux Balance Constraints (FBC) package (both in version 1 and version 2).

The COBRA Toolbox will use the provided SBML IDs as IDs for the respective elements of the model structure, and use the name fields as names. It is assumed (but not necessary), that metabolite IDs start with a "M_", reaction IDs start with a "R_" , gene IDs start with a "G_", and compartment IDs start with a "C_". This is due to the limitation on identifiers in SBML and those starting sequences will be removed if they are consistently present in the model.

Metabolite IDs of the MAT-file format use a metabolite identfier followed by a compartment identifier in square brackets (e.g. ala_L[c]). Since brackets are illegal characters for SBML IDs, The COBRA Toolbox assumes, for all non boundary species, that a compartment identifier preceded by an underscore (e.g. SBML ID: M_ala_L_c) is equivalent to the MAT-file compartment identifiers and converted to a model as such.

The COBRA Toolbox has a legacy support for the NOTE Fields defined in Schellenberger et al, Nature Protocols, 2011, but it is suggested to instead use annotations whenever possible. In general, if a fbc-package field and a NOTES field is present, the fbc-package value will be used (e.g. CHARGE for metabolites, or GENE_ASSOCIATION for reactions). The same applies to annotations, i.e. if there is an annotation for an EC number, the Notes field EC Number will be ignored. However, the charge field in SBML Level 2 will be overwritten by the Notes field definitions.

SimPheny format

SimPheny models provided in 3 or 4 files (4 if GPR rules are provided). The model identifiers will be used as presented in the SimPheny files.

Excel format

A model in a excel file formats are accepted by The COBRA Toolbox if the file adheres to the specifications listed in The COBRA Toolbox Documentation.

Available output formats

The COBRA Toolbox also allows storage in multiple file types as detailed below.

MAT-files formats

The MAT-file (`.mat`) format is most commonly used. The MAT-file format make up is a simple MATLAB `struct` containing one or more of the fields defined in The COBRA Toolbox Documentation. It has the advantage of lossless data storage even for model specific fields not supported by The COBRA Toolbox.

SBML format

SBML is a commonly used format to store biological models. The COBRA Toolbox allows the generation of models using SBML Level 3 Version 1 and uses the FBC-package extension to encode constraint based properties. This is the format that is recommended for publication, as it can be used by many different tools and allows the best use of the model.

Excel format

Historically, models were often exchanged using Excel files, and this is still in use today. Some users prefer to have an overview of a model using Excel. The COBRA Toolbox offers an Excel export of the format described in The COBRA Toolbox Documentation.

Text Format

Finally, The COBRA Toolbox offers a simple textual export, which is essentially a tab separated file containing the reactions with their reaction formulas along with the associated GPRs, but no further information. This format only uses the required fields and will ignore any optional fields.

MATERIALS

We will use two model files for this tutorial: a MAT-file formated model and an SBML formated model. First we must loads both files into the tutorial directory (cleaning any old copies).

```
cd(fileparts(which('tutorial_IO.mlx')));

% Copies the two files required for this tutorial (if they are not yet present).
try
    delete 'ecoli_core_model.mat';
    delete 'Abiotrophia_defectiva_ATCC_49176.xml'
    copyfile(which('ecoli_core_model.mat'), '.');
    copyfile(which('Abiotrophia_defectiva_ATCC_49176.xml'), '.'); %TODO: The 'Abiotrophia_defe
```

```
end
```

PROCEDURE

The time that it takes to load a model depends on the file format, the complexity of a model and the machine. The loading of a MAT-file, even of some large models, can take only seconds, whereas large SBML files can take a few minutes to load.

Reading a model (timing: 1 second to a few minutes)

The most direct way to load a model into The COBRA Toolbox is to use the `readCbModel` function. For example, to load a model from a MAT-file, you can simply use the filename (with or without file extension).

```
fileName = 'ecoli_core_model.mat';
model = readCbModel(fileName);
```

The `readCbModel` function has a second optional input that specifies the file type being loaded. In the above example the file type does not need to be specified since the input default is a 'Matlab' file type. To load file types other than a MAT-file, specificy the file type for input as: ‘SBML’, ‘SimPheny’, ‘SimPhenyPlus’, ‘SimPhenyText’, or 'Excel’.

You can also call the `readCbModel` function without a fileName to get a dialog box, this is provided the Java feature is available.

```
if usejava('desktop') % This line of code is to avoid execution of example in non gui-environm
    model = readCbModel();
end
```

Once the model is loaded it can be used directly with The COBRA Toolbox functions. To view the data stored in the model use the following command.

```
if usejava('desktop') % This line of code is to avoid execution of example in non gui-environm
    open model
end
```

Writing a model (timing: 1 second to a few minutes)

To write files, use the `writeCbModel` function. A dialog box will appear, select or enter the filename and the file format. The output is then generated and saved to the directory indicated in the diaglog box. A summary of the fields present in the model will also appear in the command window.

```
if usejava('desktop') % This line of code is to avoid execution of example in non gui-environm
    writeCbModel(model)
end
```

```
ans =
              S: [72×95 double]
           mets: {72×1 cell}
              b: [72×1 double]
         csense: [72×1 char]
           rxns: {95×1 cell}
```

```
            lb: [95×1 double]
            ub: [95×1 double]
             c: [95×1 double]
        osense: -1
         genes: {137×1 cell}
         rules: {95×1 cell}
    metFormulas: {72×1 cell}
       metNames: {72×1 cell}
    description: 'ecoli_core_model.mat'
        grRules: {95×1 cell}
     rxnGeneMat: [95×137 double]
       rxnNames: {95×1 cell}
     subSystems: {95×1 cell}
```

The `writeCbModel` function has a second optional input that specifies the file type in which the model should be written and saved. In the above example the file type was not specified and so the default file type to be saved was as a MAT-file. To use the function to write a file types other than a MAT-file, specificy the file type for input as: 'text','xls', or 'sbml'.

```
if usejava('desktop') % This line of code is to avoid execution of example in non gui-environm
    writeCbModel(model,'text')
end
```

```
ans =
            S: [72×95 double]
         mets: {72×1 cell}
            b: [72×1 double]
        csense: [72×1 char]
         rxns: {95×1 cell}
           lb: [95×1 double]
           ub: [95×1 double]
            c: [95×1 double]
        osense: -1
         genes: {137×1 cell}
         rules: {95×1 cell}
    metFormulas: {72×1 cell}
       metNames: {72×1 cell}
    description: 'ecoli_core_model.mat'
        grRules: {95×1 cell}
     rxnGeneMat: [95×137 double]
       rxnNames: {95×1 cell}
     subSystems: {95×1 cell}
```

It is also possible to specify the file type and file name explicitly. The following example writes a model directly to the file name 'Acidaminococcus.xml'.

```
if usejava('desktop') % This line of code is to avoid execution of example in non gui-environm
    writeCbModel(model, 'SBML', 'Acidaminococcus.xml')
end
```

```
Document written
ans =
        constraint: [1×0 struct]
    functionDefinition: [1×0 struct]
             event: [1×0 struct]
              rule: [1×0 struct]
        unitDefinition: [1×1 struct]
```

```
  initialAssignment: [1×0 struct]
         SBML_level: 3
       SBML_version: 1
         annotation: ''
          areaUnits: ''
    avogadro_symbol: ''
     conversionFactor: ''
        delay_symbol: ''
         extentUnits: ''
  fbc_activeObjective: 'obj'
        fbc_version: 2
                 id: 'COBRAModel'
        lengthUnits: ''
             metaid: 'COBRAModel'
               name: ''
              notes: ''
            sboTerm: -1
     substanceUnits: ''
          timeUnits: ''
        time_symbol: ''
           typecode: 'SBML_MODEL'
        volumeUnits: ''
            species: [1×72 struct]
        compartment: [1×2 struct]
          parameter: [1×5 struct]
           reaction: [1×95 struct]
      fbc_fluxBound: [1×2 struct]
    fbc_geneProduct: [1×137 struct]
      fbc_objective: [1×1 struct]
         namespaces: [1×2 struct]
         fbc_strict: 1
```

CLEAN UP

Clean up of materials used in the tutorial.

```
currentDir = pwd;
cd(fileparts(which('tutorial_IO.mlx')));

% Delete the files used in this tutorial (if they are present).
try
    delete('ecoli_core_model.mat');
    delete('Abiotrophia_defectiva_ATCC_49176.xml'); %TODO: The 'Abiotrophia_defectiva_ATCC_491
    delete('Acidaminococcus.xml');
end
cd(currentDir)
```