IO Tutorial

Author(s): Thomas Pfau, University of Luxembourg

Reviewers:

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. Internally, it uses a simple MATLAB `struct` with fields defined in the Documentation. These models are commonly provided in a `.mat` file.

There are additional model formats for which io functions exist in The COBRA Toolbox. These incluse

- Models in SBML Format
- Models in SimPheny format
- Models in an Excel Format

MATLAB save files:

The format of a model struct provided in a `.mat` file has to stick to the rules defined in the Documentation.

SBML Format

The COBRA Toolbox currently supports SBML models provided as Level 3 version 1 (as defined here) and has legacy support for older versions of SBML. It also supports the Level 3 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. Since metabolites in The COBRA Toolbox model `struct` are commonly provided with a metabolite ID followed by a compartment identifier e.g. ('ala_L[c]'), and brackets are illegal characters for an SBML ID. It is assumed, that if all non boundary species have a trailing compartment identifier preceded by a underscore (e.g. SBML ID: M_ala_L_c) those identifiers should be converted to model compartment identifiers.

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 provide their models 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

Excel files adhereing to the COBRA xls specifications listed in the Documentation. Parseable Excel models contain two sheets for reactions and metabolites respectively.

Available output formats

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

MATLAB `.mat` files

This format is simply the model being saved as a MATLAB save file, and is the default save method. It has the advantage of lossless data storage even for model specific fields not supported by The COBRA Toolbox.

SBML format (`.xml`)

The systems biology markup language (SBML) is a very common format to store biological models. The COBRA Toolbox allows generation of models using 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 get an overview over a model using Excel, since it provides a general overview at a first glance. The COBRA Toolbox offers an Excel export using the format described in the Documentation (see above).

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, one MATLAB `.mat` file and an `.xml` file stored in SBML format. The following code loads these files into the tutorial directory (cleaning any old copies).

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

% Copy two files that can be loaded (if they are nto 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'), '.');
end
```

```
Warning: File 'ecoli_core_model.mat' not found.
Warning: File 'Abiotrophia_defectiva_ATCC_49176.xml' not found.
```

PROCEDURE

The time it takes to load a model depends on the file format and the complexity of a model. While `.mat` file loading even for large models is a question of seconds, very large SBML files can take a few minutes to load depending on the machine.

Reading a model (TIMING 1s to a few minutes)

The most straightforward way to import a model into the The COBRA Toolbox is to use the `readCbModel` function. To e.g. load a model from a `.mat` file, you can simply use the filename (with or without file extension).

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

`readCbModel` assumes, that .mat files are a MATLAB save file, `.xml` files are models in SBML format, `.sto` are SimPheny models, and `.xls` or `.xlsx` are models in Excel format.

```
% This code is to avoid execution in non gui-environments
if usejava('desktop')
```

You can also call the function without a fileName to get a file selection dialog

```
model = readCbModel();
```

The model loaded in this way can directly be used with The COBRA Toolbox functions. You can view the data stored in the model by e.g. using

```
open model

% This code is to avoid execution in non gui-environments
end
```

Writing a model (TIMING: 1s to a few minutes)

```
% This code is to avoid execution in non gui-environments
if usejava('desktop')
```

To write files, use the `writeCbModel` function. The function can be called directly with a model.

```
writeCbModel(model)
```

This will call a file selection dialog, which allows the selection of a filename and, depending on the selected format from the dropdown, the output will be generated.

```
% This code is to avoid execution in non gui-environments
end
```

```
% This code is to avoid execution in non gui-environments
if usejava('desktop')
```

If no `fileName` is provided, a popup will ask you to provide a `fileName` with the specified format.

```
writeCbModel(model,'text')
```

The available format options are:

- `'mat'` - for a MATLAB save file
- `'sbml'` - for a SBML model
- `'xls'` - for a model in Excel format
- `'text'` - for a textual export.

```
% This code is to avoid execution in non gui-environments
end
```

It is also possible to specify the format and filename explicitly:

```
writeCbModel(model, 'SBML', 'Acidaminococcus.xml')
```

```
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
```

which will write the model to the file Acidaminococcus.xml.

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

% Copy two files that can be loaded (if they are nto yet present).
try
    delete('ecoli_core_model.mat');
    delete('Abiotrophia_defectiva_ATCC_49176.xml');
    delete('Acidaminococcus.xml');
end
cd(currentDir)
```