SAMMIM Documentation

Andre Schultz

Apr 07, 2020

Contents:

SAMMIM is a tool for visualizing metabolic networks and metabolic network simulations using SAMMI directly from MATLAB using the COBRA toolbox. This documentation describes the MATLAB wrapper for this visualization. You can view the full documentation for SAMMI here, and the documentation for COBRA here.

If you use SAMMI for your project, please cite the following publication: Schultz, A., & Akbani, R. (2019). SAMMI: A Semi-Automated Tool for the Visualization of Metabolic Networks. *Bioinformatics*.

CHAPTER 1

Installation and Usage

To use this tool simply add the SAMMIM folder to your MATLAB path. There is no need to add its subfolders. For a short description of how to use the plotting function type:

```
help sammi
```

Some of the functionality available in SAMMI, such as PDF download and data upload, are not directly available though this plugin. To use these functions download the model in a SAMMI format and upload the file in the SAMMI web interface at www.SammiTool.com.

For a full description of this plugin please refer to the remaining sections of this documentation.

CHAPTER 2

Documentation

2.1 Structure Arrays

Three different structure arrays can be used to render visualizations using the SAMMIM package. To learn more about MATLAB structure arrays please refer to this documentation. This section describes these variables. For details on how to use them and working examples please refer to the subsequent sections. These variables are the following:

2.1.1 Parser

Using SAMMI, models can be parsed into subgraphs upon rendering for more appealing visualizations. While there are many different ways to parse models (covered in the next section), one of the options is defining a structure array of length `n` where a subgraph is plotted for each element of the array. Each element in the array should contain the following fields:

- **rxns**: Cell array of strings. Reaction IDs of reactions to be included in the subgraph.
- **name**: String. Name of the subgraph to be displayed in the visualization.
- **flux**: Numerical vector. Optional. Values to be mapped as reaction colors. Defaults to all NaNs where no data is mapped.

2.1.2 Data

The data structured array can be used to map different datasets, in different forms, onto the rendered illustration. This structured array can also be of length `n`, where each element of the array will map a dataset differently. Each element of the array must contain the following fields:

- **type**: Cell array of two strings. The first string should be either `rxns`, `mets`, or `links`. This string defines where the data will be mapped. The second string should be either `color` or `size`. This string defines what kind of data to map onto the defined group. `color` can be user with `rxns` and `mets` to define node and link color. `color` cannot be used with `links`, as link color matches the corresponding reaction node color. `size` can be used with all options to define node radius or link width.

- **data**: A MATLAB table object. To learn more about MATLAB tables please refer to this documentation. `VariableNames` (column names) will be translated into condition names, and `RowNames` should be reaction IDs for `rxns` and `links` data or metabolite IDs for `mets` data. `NaN` values will not be mapped.

2.1.3 Options

Optional structured array to change default rendering options. This is an array of length one with three optional fields:

- **htmlName**: String. Name of html file where the output will be written. Defaults to `index_load.html`. If this option is not defined, the file `index_load.html` will be continuously overwritten every time a new visualization is generated. If users wish to save a visualization to a different file, or wish to visualize multiple maps at once, this parameter can be changed.
- **load**: Boolean, defaults to `True`. Whether or not to load the visualization on a new browser tab. If this parameter is set to false, new visualizations can be rendered by refreshing a previously loaded tab or by using the `openSammi()` function.
- **jscode**: String. Sequence of JavaScript commands to be run following the rendering of the visualization. This can be used, for example, to change colorscales and subgraphs upon loading the model. This options requires familiarity with JavaScript and the SAMMI html layout.

2.2 Functions

There are three main user functions for rendering and plotting SAMMI visualizations. These are:

2.2.1 Plotting

The function `sammi` is used to plot SAMMI visualizations in combination with the SAMMI structured arrays. The function definition is:

```
sammi(model,parser,data,secondaries,options)
```

The function inputs are as follows:

- **model**: COBRA model to be visualized.
- **parser**: How the model is to be parsed and visualized. There are several options for this parameter:
  - *Empty vector*: Default. Does not parse the model and plots all reactions and metabolites in a single graph. Not recommended for medium to large-size models.
  - *string*: One of two options. (1) A reaction or metabolite field (e.g. `subsystem` or `compartment`), in which case a subgraph will be drawn for each unique value associated with that field. (2) A path to a file specifying a previously drawn SAMMI map, in which case that map will be rendered.
  - *Cell array of strings*: List of reaction IDs to be plotted. A single graph will be plotted containing only the defined reactions.
  - *Parser structured array of length n*: A subgraph is plotted for each element of the array as defined in the previous section.
- **data**: Data structured array of length `n`. All defined datasets and data types will be mapped as described in the previous section.

- **secondaries**: Cell array of strings or regular expressions. Any metabolite, in any subgraph, matching any of the regular expressions defined here will be shelved. These metabolites are not deleted and can be returned to the graph through the floating menu window. For details of this functionality please refer to the SAMMI documentation.
- **options**: Options structured array as defined in the previous section. Additional options for loading the map.

2.2.2 Opening a visualization

The function `openSammi` is used to open previously drawn visualizations. It takes a single input: a string defining a previously drawn map html file name. For instance, `openSammi('index_load.html')` or `openSammi('index_load')` opens the default html file to which SAMMIM writes metabolic maps.

2.2.3 Running SAMMIM examples

The SAMMIM examples described in the next section are available in the `testSammi` function. To try the described examples run `testSammi(n)` where `n` ranges from zero to eleven referring to the example number. To visualize the code for each example run `edit testSammi` from the MATLAB command window.

2.3 Examples

Here we provide several simple examples for the use of SAMMIM. Each example is supposed to be more complex than the next, and is intended to exemplify as many different functionalities of SAMMIM as possible. To get started, install the COBRA toolbox as described in the documentation here. The examples described here are available through SAMMIM in the `testSammi` function. To view the code for each example in MATLAB type `edit testSammi`. To run each example run `testSammi(n)` where `n` ranges from zero to eleven, referring to the example number defined here.

Make sure the COBRA toolbox is initialized and load variable to be used throughout these examples:

```
%Initialize toolbox
initCobraToolbox
%Get COBRA directory
global CBTDIR;
%Get SAMMIM folder
sammipath = strrep(which('sammi'),'\sammi.m','');
```

2.3.1 0. Plot entire model

To plot the entire model simply call `sammi` on the COBRA model. This is not advisable for medium to large models as the visualization may be too large to render.

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Plot
sammi(model)
```

2.3.2 1-2. Divide the model into subgraphs using model annotation

1. Maps can be divided into subgraphs using model annotation. For instance, users can plot a subgraph for each annotated reaction subsystem:

```
%Load model
load([CBTDIR '/test/models/mat/Recon2.v04.mat'])
%Plot a subgraph for each subsystem
sammi(modelR204,'subSystems')
```

2. Or plot a map for each cellular compartment:

```
%Get sample model to plot
load([CBTDIR '/test/models/mat/ecoli_core_model.mat']);
%Make compartment vector
comp = regexp(model.mets,'\[(.)\]$','tokens');
comp = vertcat(comp{:});
model.compartment = vertcat(comp{:});
%Plot
sammi(model,'compartment')
```

2.3.3 3. Plot and visualize multiple maps

By default, SAMMI outputs the visualization to a file names `index.load.html` in the package folder. Therefore, by default, every time a new visualization is generated this file is overwritten. The name of the output file can be changed, however, in order to not overwrite files. For instance:

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Initialize options
options.load = false;
%Plot whole model in default file
sammi(model,[],[],[],options)
%Modify options to render map in different file
options.htmlName = 'index_load2.html';
%Plot model parsed by subsystem
sammi(model,'subSystems',[],[],options)
%Open both visualizations
openSammi('index_load.html')
openSammi('index_load2.html')
```

2.3.4 4. Plot only user-defined reactions

For a quick visualization of a given group of reactions users can plot only certain reactions in a single graph.

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Get reactions to plot
tca = {'ACONTa','ACONTb','AKGDH','CS','FUM','ICDHyr','MDH','SUCOAS'};
gly = {'ENO','FBA','FBP','GAPD','PDH','PFK','PGI','PGK','PGM','PPS','PYK','TPI'};
ppp = {'G6PDH2r','GND','PGL','RPE','RPI','TALA','TKT1','TKT2'};
dat = cat(2,tca,gly,ppp);
%Plot only desired reactions
sammi(model,dat);
```

2.3.5 5. Shelve secondary metabolites on load

In order to shelve secondary metabolites upon rendering the model, define the `secondaries` input to the plot function. If this argument is defined, any metabolite, matching any of the defined regular expressions, will be shelved. These metabolites can be returned to the graph using the floating menu window.

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Get reactions to plot
tca = {'ACONTa','ACONTb','AKGDH','CS','FUM','ICDHyr','MDH','SUCOAS'};
gly = {'ENO','FBA','FBP','GAPD','PDH','PFK','PGI','PGK','PGM','PPS','PYK','TPI'};
ppp = {'G6PDH2r','GND','PGL','RPE','RPI','TALA','TKT1','TKT2'};
dat = cat(2,tca,gly,ppp);
%Define secondaries
secondaries = {'^h\[.\]$','^h2o\[.\]$','^o2\[.\]$','^co2\[.\]$',...
    '^atp\[.\]$','^adp\[.\]$','^pi\[.\]$',...
    '^nadh\[.\]$','^nadph\[.\]$','^nad\[.\]$','^nadp\[.\]$'};
%Plot only desired reactions
sammi(model,dat,[],secondaries);
```

2.3.6 6. Plot multiple user-defined subgraphs

Users can also plot multiple subgraphs with their defined reactions. To do so, define the Parser structured array for each subgraph:

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Get reactions to plot
dat = struct;
dat(1).name = 'TCA Cycle';
dat(1).rxns = {'ACONTa';'ACONTb';'AKGDH';'CS';'FUM';'ICDHyr';'MDH';'SUCOAS'};
dat(2).name = 'Glycolysis';
dat(2).rxns = {'ENO';'FBA';'FBP';'GAPD';'PDH';'PFK';'PGI';'PGK';'PGM';'PPS';'PYK';'TPI
↪'};
dat(3).name = 'Pentose Phosphate Pathway';
dat(3).rxns = {'G6PDH2r';'GND';'PGL';'RPE';'RPI';'TALA';'TKT1';'TKT2'};
%Plot only desired reactions
sammi(model,dat);
```

2.3.7 7-8. Data mapping

7. Add data to plotted subgraphs. In this example we are generating random data and mapping it onto the desired reactions. Using the Parser structured array users can directly map data as reaction colors:

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Get reactions to plot
dat = struct;
dat(1).name = 'TCA Cycle';
dat(1).rxns = {'ACONTa';'ACONTb';'AKGDH';'CS';'FUM';'ICDHyr';'MDH';'SUCOAS'};
dat(2).name = 'Glycolysis';
dat(2).rxns = {'ENO';'FBA';'FBP';'GAPD';'PDH';'PFK';'PGI';'PGK';'PGM';'PPS';'PYK';'TPI
↪'};
dat(3).name = 'Pentose Phosphate Pathway';
```

(continues on next page)

(continued from previous page)

```
dat(3).rxns = {'G6PDH2r';'GND';'PGL';'RPE';'RPI';'TALA';'TKT1';'TKT2'};
%Add random flux
for i = 1:3; dat(i).flux = randn(length(dat(i).rxns),1); end
%Plot only desired reactions
sammi(model,dat);
```

8. Alternatively, users can map data onto the map using the Data structured array. The following example maps five sets of random data, each in a different way, with five conditions each.

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Define number of conditions
n = 5;
%Make reaction table with random data
rxntbl = randn(length(model.rxns),n);
rxntbl(randsample(length(model.rxns)*n,floor(n*length(model.rxns)/10))) = NaN;
rxntbl = array2table(rxntbl,'VariableNames',sprintfc('condition_%d',1:n),...
    'RowNames',model.rxns);
%Make metabolites table with random data
mettbl = randn(length(model.mets),n);
mettbl(randsample(length(model.mets)*n,floor(0.5*length(model.mets)))) = NaN;
mettbl = array2table(mettbl,'VariableNames',sprintfc('condition_%d',1:n),...
    'RowNames',model.mets);
%Make struct
dat = struct;
dat(1).type = {'rxns' 'color'};
dat(1).data = rxntbl;
dat(2).type = {'rxns' 'size'};
dat(2).data = rxntbl;
dat(3).type = {'mets' 'color'};
dat(3).data = mettbl;
dat(4).type = {'mets' 'size'};
dat(4).data = mettbl;
dat(5).type = {'links' 'size'};
dat(5).data = rxntbl;
%Define secondaries
secondaries = {'^h\[.\]$','^h20\[.\]$','^o2\[.\]$','^co2\[.\]$',...
    '^atp\[.\]$','^adp\[.\]$','^pi\[.\]$',...
    '^nadh\[.\]$','^nadph\[.\]$','^nad\[.\]$','^nadp\[.\]$'};
%Plot dividing up by subsystems
sammi(model,'subSystems',dat,secondaries)
```

2.3.8 9. Change map upon load

SAMMI options also allow users to change visualization parameters upon loading the model. This can be done by adding JavaScript code to the end of the visualization. To use this advanced feature users need to be familiar with JavaScript and need to familiarize themselves with the SAMMI visualization html layout. The following code loads the previous map, changes the visualization to the Citric Acid Cycle subgraph, and changes the colorscale upon loading.

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Define number of conditions
n = 5;
```

(continues on next page)

(continued from previous page)

```
%Make reaction table with random data
rxntbl = randn(length(model.rxns),n);
rxntbl(randsample(length(model.rxns)*n,floor(n*length(model.rxns)/10))) = NaN;
rxntbl = array2table(rxntbl,'VariableNames',sprintfc('condition_%d',1:n),...
    'RowNames',model.rxns);
%Make struct
dat = struct;
dat(1).type = {'rxns' 'color'};
dat(1).data = rxntbl;
%Define secondaries
secondaries = {'^h\[.\]$','^h20\[.\]$','^o2\[.\]$','^co2\[.\]$',...
    '^atp\[.\]$','^adp\[.\]$','^pi\[.\]$',...
    '^nadh\[.\]$','^nadph\[.\]$','^nad\[.\]$','^nadp\[.\]$'};
%Define Javascript code
jscode = ['x = document.getElementById("onloadf1");' ...
    'x.value = "Citric Acid Cycle";' ...
    'onLoadSwitch(x);' ...
    'document.getElementById("fluxmin").valueAsNumber = -0.1;' ...
    'document.getElementById("fluxmax").valueAsNumber = 0.1;' ...
    'fluxmin = -0.1; fluxmax = 0.1;' ...
    'document.getElementById("edgemin").value = "#ff0000";' ...
    'document.getElementById("edgemax").value = "#0000ff";' ...
    'document.getElementById("addrxnbreak").click();' ...
    'document.getElementsByClassName("rxnbreakval")[0].value = 0;' ...
    'document.getElementsByClassName("rxnbreakcol")[0].value = "#c0c0c0";' ...
    'defineFluxColorVectors();'];
%Define options
options.jscode = jscode;
%Plot dividing up by subsystems
sammi(model,'subSystems',dat,secondaries,options)
```

2.3.9 10. Load Existing Map

SAMMI makes it easy for users to share curated maps through the SAMMI Json export. To load existing maps, pass the file path to the `parser` argument. The following example load a map included with the SAMMIM folder:

```
%Load model
load([CBTDIR '/test/models/mat/ecoli_core_model.mat'])
%Define zooming option
options.jscode = 'zoom.transform(gMain, d3.zoomIdentity.translate(-1149,-863).scale(2.
↪64));';
%Load existing model
sammi(model,[sammipath '\demo.json'],[],[],options)
```

2.3.10 11. Type-III Pathways

Type-III pathways are thermodynamically infeasible, flux-balanced distributions that do not include exchange reactions. In this example we use SAMMI to visualize type-III pathways in the iJO1366 model. We first block all exchange reactions, then perform FVA to determine which reactions remain active. We then loop through the active reactions using FBA to determine loops where they are active.

```
%Load and tailor model
load([CBTDIR '/test/models/mat/iJO1366.mat'])
```

(continues on next page)

(continued from previous page)

```
model = iJO1366;
model = changeRxnBounds(model,model.rxns(findExcRxns(model)),0,'b');
model = changeRxnBounds(model,'ATPM',0,'l');
model.csense = repmat('E',length(model.mets),1);
model.c = model.c*0;

%Do FVA
[fluxmin,fluxmax] = fastFVA(model,0);
%Clear numerical error
fluxmax(fluxmax < 1e-7) = 0;
fluxmin(fluxmin < -1e-7) = 0;

%Parse
count = 0;
%For each positive flux
dat = struct;
for id = find(fluxmax)'
    %Set as objective
    model = changeObjective(model,model.rxns{id},1);
    %Calculate fluxes
    flux = optimizeCbModel(model,'max','one');
    %Clear numerical error
    flux.x(abs(flux.x) < 1e-7) = 0;
    %Save results for plot
    count = count+1;
    ind = find(flux.x);
    dat(count).name = num2str(count);
    dat(count).rxns = model.rxns(ind);
    dat(count).flux = flux.x(ind);
end
%For each negative flux
for id = find(fluxmin)'
    %Set as objective
    model = changeObjective(model,model.rxns{id},1);
    %Calculate fluxes
    flux = optimizeCbModel(model,'min','one');
    %Clear numerical error
    flux.x(abs(flux.x) < 1e-7) = 0;
    %Save results for plot
    count = count+1;
    ind = find(flux.x);
    dat(count).name = num2str(count);
    dat(count).rxns = model.rxns(ind);
    dat(count).flux = flux.x(ind);
end
%Plot
sammi(model,dat)
```