Small tutorial on the archive object for whole cell model organization

Rick Vink

April 19, 2016

Keywords: Whole cell model, *Mycoplasma genitalium*, MATLAB, organizing, tutorial.

1 Introduction

All methods should be called like $'archive.method'$ as shown in example 1. I leave this out in most example because it just takes space.

$$>> archive.method \tag{1}$$

Almost all my methods have a help function. This can be called as shown in example 2, where method corresponds to one of the methods. The help function includes some explanation and many additional options which I do not cover in this report.

$$>> help \quad archive.method \tag{2}$$

2 Initiation archive object

The first step of creating an new archive is to initiate the object (example (7)). This step does not involve any operations on simulations. It includes to direct to the right directory, creating a new archive object and retrieving general information about the whole cell model.

$$\begin{aligned} &>> cd \quad '/home/path/to/Archive' \\ &>> archive = Archive; \\ &>> archive.general\_information; \end{aligned} \tag{3}$$

The command $general\_information$ includes multiple sub methods that extract data from multiple different xml files. These xml files are stored inside the folder "data_base".

The result is an object that contains four properties. One of the properties contains the data sets of the simulation and is denoted as $'set'$. $'info'$ contains the information about the whole cell model; including the labels corresponding to elements of the matrix inside state files. $'settings'$ is used to store information about the archive. This ranges from settings used during certain analysis steps and personal preferences. The fourth property is $'stoichiometry'$. This property will probably not be used most of the time. It contains the stoichiometric relationships of the model which has been used during the construction of the metabolic pathway visualization.

$$\begin{aligned} &archive. \\ &\quad set : [struct] \\ &\quad info : [struct] \\ &\quad settings : [struct] \\ &\quad stoichiometry : [struct] \end{aligned} \tag{4}$$

3 Add simulation data

Storing the paths of different simulations files and their properties is the main objective of the archive object. New data can be added by calling the method $'add\_sets'$ (example 5). This function only requires the path to the folder which contains the simulation data, and searches for simulations regardless of the folder structure. The only requirement is that all the files belonging to a simulation are present in one folder. This allows you to add all the simulations present on a hard drive with a single command. The function belonging to the method searches through all the subfolders and their subfolders and so on, to find $'options.mat'$ files. These $'options.mat'$ are used to extract information about the single gene knockouts and the seeds of the simulations.

$$add\_sets('/home/path/to/simulation/dir'); \tag{5}$$

The simulations are divided into different sets. At this moment of time, the simulations are only discriminated by their knockouts. This results in the creation of a new set every time a simulations with a different set of knockouts is added to the archive. In example 6 is illustrated how the archive will look like after adding sets in which three simulations share the same set of knockouts, same holds for two other simulations and one simulations has its own set of knockouts.

$archive.$
$set(1).simulation(1).folder =' path/folder1';$
$set(1).simulation(2).folder =' path/folder2';$
$set(1).simulation(3).folder =' path/folder3';$ (6)
$set(2).simulation(1).folder =' path/folder4';$
$set(3).simulation(1).folder =' path/folder5';$
$set(3).simulation(2).folder =' path/folder6';$

This process can take quite a long time in the case that there are many different files and sub-folders. It might be worthwhile to add simulations in smaller bashes. For example, adding 500 simulations at the time instead of 4000 in one go by scanning a complex folder structure.

The archive can be checked for non-unique simulations using the method $'check\_unique\_simulations'$. This function compares the seeds within every set and labels a copy of a simulations as non-unique. This is handy when accidentally adding a set to the archive twice or in the case when the seeds of the simulations are not set properly. subsequently, the non-unique simulations can be removed using the method $'remove_n on_u nique_s imulations'$.

4 Compression

Working with the raw data of the simulations requires a lot of computational power and take a large storage space. This is due to the fact that the state of a cell is saved for every second of the simulation. Thereby, a simulation of a wild type cell generates generally 300 MB of data. This data is saved in multiple files in which 100 data points are saved in each file. The functions of the archive therefore use a compressed version of the raw data on default.

A compression method called $'compress\_state\_files'$ is implemented for the archive which reduces the multiple state files into two different state files. One of these files contains the mean value of every state file and the other file contains the last value of every state file. These files have a size of about 2 a 4 MB respectively. This method can be performed on an external cluster in order to reduce the size of the data before downloading it to a local computer.

5 Basic processing

A logical next step is to execute the methods $'general\_processing'$. This method includes methods that are required for many other methods.

6 Examples

Let's do some examples what you can do using this archive.

6.1 Initiate

Multiple steps to create general archive structure. This structure is required for all other examples

$$\begin{aligned}
&>> cd \quad '/home/path/to/Archive'\\
&>> archive = Archive;\\
&>> archive.general_information;\\
&>> archive.add_sets('path/to/dir');\\
&>> archive.compress_state_files;\\
&>> archive.general_processing;
\end{aligned} \tag{7}$$

6.2 Plotting

Plot the concentration of 'ATP' of the first set inside the cytosol.

$$\begin{aligned}
&>>M =' ATP';\\
&>>C = 1; \quad \%Cytosol\\
&>>set = 1;\\
&>>N = find(strcmpi(...\\
&\quad archive.info.Metabolite.name, M)));\\
&>>archive.plot(set, ...\\
&\quad \{'Metabolite','counts'\}, N, C);
\end{aligned} \tag{8}$$

Plot the flux value of the metabolic reaction Aas1 of the 9th simulation of the 5th set. And save it afterwards into the default folder $'output'$ in the default formats $'.pdf'$, $'.fig'$ and $'.png'$ as $'Aas1\_flux'$.

$$\begin{aligned}
&>>I =' Aas1';\\
&>>set = 5;\\
&>>N = find(strcmpi(...\\
&\quad archive.info.MetabolicReaction.ID, I)));\\
&>>archive.plot(set, ...\\
&\quad \{'MetabolicReaction','fluxs'\}, N, 1, ...\\
&\quad 'simulation', 5);\\
&>>archive.save_figure('Aas1_flux');
\end{aligned} \tag{9}$$

6.3 Open file

Load a $'.mat'$ file of the simulation. For example the 5th state file of simulation 3 of set 1.

$$\begin{aligned}
&>>set = 1;\\
&>>sim = 3;\\
&>>file = \quad 'state - 5.mat';\\
&>>state5 = ...\\
&\quad archive.load_file(set, sim, ...\\
&\quad 'fileTypeTag', file);
\end{aligned} \tag{10}$$

6.4 Inspect data

Check and remove non unique and crashed simulations

```
>>archive.check_unique_simulations;
>>archive.remove_non_unique_simulations
>>archive.check_crashed_simulations;
>>archive.remove_crashed_simulations
```
(11)

Check which sets have less than 5 simulations.

```
>>mSize = 5;
>>tSet = archive.sets;
>>for  iSet = 1 : tSet
>>tSim = archive.simulations(iSet);
>>if  tSim < mSize
>>fprintf('set%d/n');
>>end
>>end
```
(12)

6.5 Dendrogram

Make a dendrogram at time point 25000 seconds using the concentration values of the metabolites.

```
>>time = 25000;
>>compartments = 1 : 6;
>>archive.snap_shot(time, ...
    'Metabolite','counts', ...
    'compartment', compartments);
>>ss = 1; %  selectsnap_shot
>>archive.dendrogram(ss);
```
(13)

6.6 Metabolic pathway visualization

Create a visual representation of the metabolic pathway of data set 2 at time point 25000. Set 1 (wild type cell) is used as reference to normalize the values of set 2.

```
>>time = 25000;
>>ref = 1;
>>comp = 2;
>>tmp = archive.average_value(...
    ref, time,  'Metabolite',  'counts', 1 : 6))
>>met_ref = squeeze(tmp);
>>tmp = archive.average_value(...
    comp, time,  'Metabolite',  'counts', 1 : 6)
>>met_target = squeeze(tmp);
>>react_ref = archive.average_value(...
    ref, time,' MetabolicReaction',' fluxs', 1)
>>react_target = archive.average_value(
    comp, time,' MetabolicReaction',' fluxs', 1)
>>met = 1 − ((met_target − met_ref)/...
    met_ref + 0.5);
>>react = 1 − ((react_target − react_ref)/...
    react_ref + 0.5);
>>archive.cyto_viz(  'Metabolite',  'On', ...
    'metaboliteData', met,
    'MetabolicReaction',  'On', ...
    'colorData', react);
```
(14)