Create an overview table with model properties

Author(s): Ines Thiele, Ronan M. T. Fleming, Systems Biochemistry Group, LCSB, University of Luxembourg.

Reviewer(s): Catherine Fleming, Stefania Magnusdottir, Molecular Systems Physiology Group, LCSB, University of Luxembourg.

INTRODUCTION

In this tutorial, we evaluate the basic properties of the metabolic model, such as the number of reactions, unique metabolites, blocked reactions, dead-end metabolites, and store the information in a table ('Table_Prop').

EQUIPMENT SETUP

Initialize the COBRA Toolbox.

If necessary, initialize The Cobra Toolbox using the `initCobraToolbox` function.

```
initCobraToolbox
```

```
      _____   _____   _____   _____     _____     |
     / ___| / _   \ |  _  \ |  _  \   / ___ \    |   COnstraint-Based Reconstruction and Analysis
    | |     | | | | | |_| | | |_| |  | |___| |   |   The COBRA Toolbox - 2017
    | |     | | | | |  _  { |  _  /  |  ___  |   |
    | |___  | |_| | | |_| | | | \ \  | |   | |   |   Documentation:
     \____| \_____/ |_____/ |_|  \_\ |_|   |_|   |   http://opencobra.github.io/cobratoolbox
                                                  |

 > Checking if git is installed ...  Done.
 > Checking if the repository is tracked using git ...  Done.
 > Checking if curl is installed ...  Done.
 > Checking if remote can be reached ...  Done.
 > Initializing and updating submodules ... Done.
 > Adding all the files of The COBRA Toolbox ...  Done.
 > Define CB map output... set to svg.
 > Retrieving models ...   Done.
 > TranslateSBML is installed and working properly.
 > Configuring solver environment variables ...
   - [---*] ILOG_CPLEX_PATH: C:\Program Files\IBM\ILOG\CPLEX_Studio1271\cplex\matlab\x64_win64
   - [----] GUROBI_PATH :  --> set this path manually after installing the solver ( see instructions )
   - [---*] TOMLAB_PATH: C:\Program Files\tomlab\
   - [----] MOSEK_PATH :  --> set this path manually after installing the solver ( see instructions )
   Done.
 > Checking available solvers and solver interfaces ... Done.
 > Setting default solvers ... Done.
 > Saving the MATLAB path ... Done.
   - The MATLAB path was saved in the default location.

 > Summary of available solvers and solver interfaces

     Support           LP   MILP     QP   MIQP    NLP
 ----------------------------------------------------------------------
 cplex_direct  full              0      0      0      0      -
 dqqMinos      full              0      -      -      -      -
 glpk          full              1      1      -      -      -
 gurobi        full              1      1      1      1      -
 ibm_cplex     full              1      1      1      -      -
 matlab        full              1      -      -      -      1
 mosek         full              0      0      0      -      -
 pdco          full              1      -      1      -      -
 quadMinos     full              0      -      -      -      0
 tomlab_cplex  full              1      1      1      1      -
 qpng          experimental      -      -      1      -      -
 tomlab_snopt  experimental      -      -      -      -      1
 gurobi_mex    legacy            0      0      0      0      -
 lindo_old     legacy            0      -      -      -      -
 lindo_legacy  legacy            0      -      -      -      -
 lp_solve      legacy            1      -      -      -      -
 opti          legacy            0      0      0      0      0
 ----------------------------------------------------------------------
 Total         -                 7      4      5      2      2

 + Legend: - = not applicable, 0 = solver not compatible or not installed, 1 = solver installed.


 > You can solve LP problems using: 'glpk' - 'gurobi' - 'ibm_cplex' - 'matlab' - 'pdco' - 'tomlab_cplex' - 'lp_solve'
 > You can solve MILP problems using: 'glpk' - 'gurobi' - 'ibm_cplex' - 'tomlab_cplex'
 > You can solve QP problems using: 'gurobi' - 'ibm_cplex' - 'pdco' - 'tomlab_cplex' - 'qpng'
 > You can solve MIQP problems using: 'gurobi' - 'tomlab_cplex'
 > You can solve NLP problems using: 'matlab' - 'tomlab_snopt'

 > Checking for available updates ...
 --> You cannot update your fork using updateCobraToolbox(). [3d2698 @ Tutorial-modelProperties].
     Please use the MATLAB.devTools (https://github.com/opencobra/MATLAB.devTools).
```

Setting the optimization solver.

This tutorial will be run with a `'glpk'` package, which is a linear programming (`'LP'`) solver. The `'glpk'` package does not require additional instalation and configuration.

```
solverName='glpk';
solverType='LP';
changeCobraSolver(solverName,solverType);
```

However, for the analysis of large models, such as Recon 2.04, it is not recommended to use the 'glpk' package but rather an industrial strength solver, such as the 'gurobi' package. For detailed information, refer to The Cobra Toolbox solver instalation guide.

A solver package may offer different types of optimization programmes to solve a problem. The above example used a LP optimization, other types of optimization programmes include; mixed-integer linear programming ('MILP'), quadratic programming ('QP'), and mixed-integer quadratic programming ('MIQP').

```
warning off MATLAB:subscripting:noSubscriptsSpecified
```

COBRA model.

In this tutorial, the model used is the generic reconstruction of human metabolism, the Recon 2.04 [1], which is provided in the COBRA Toolbox. The Recon 2.04 model can also be downloaded from the Virtual Metabolic Human webpage. Before proceeding with the simulations, the path to the model needs to be set up and the model loaded:

```
modelFileName = 'Recon2.v04.mat';
modelDirectory = getDistributedModelFolder(modelFileName); %Look up the folder for the distributed Models.
modelFileName= [modelDirectory filesep modelFileName]; % Get the full path. Necessary to be sure, that the right model is loaded
model = readCbModel(modelFileName);
```

PROCEDURE

We first initialize the table

```
clear TableProp
r = 1;
TableProp(r, :) = {'Model'}; r = r+1;
```

Determine the number of reactions in the model.

```
TableProp(r, 1) = {'Reactions'};
TableProp{r, 2} = num2str(length(model.rxns));
r = r + 1;
```

Determine the number of metabolites in the model.

```
TableProp(r, 1) = {'Metabolites'};
TableProp{r, 2} = num2str(length(model.mets));
r = r + 1;
```

Determine the number of unique metabolites in the model.

```
TableProp(r, 1) = {'Metabolites (unique)'};
[g, remR3M] = strtok(model.mets,'[');
TableProp{r, 2} = num2str(length(unique(g)));
r = r + 1;
```

Determine the number of compartments in model.

```
TableProp(r, 1) = {'Compartments (unique)'};
TableProp{r, 2} = num2str(length(unique(remR3M)));
r = r + 1;
```

Determine the number of unique genes.

```
TableProp(r, 1) = {'Genes (unique)'};
[g,rem]=strtok(model.genes,'.');
TableProp{r, 2} = num2str(length(unique(g)));
r = r + 1;
```

Determine the number of subsystems.

```
TableProp(r, 1) = {'Subsystems'};
TableProp{r, 2} = num2str(length(unique(model.subSystems)));
r = r + 1;
```

Determine the number of deadends.

```
TableProp(r, 1) = {'Deadends'};
D3M = detectDeadEnds(model);
TableProp{r, 2} = num2str(length(D3M));
r = r + 1;
```

Determine the size of the S matrix.

```
TableProp(r, 1) = {'Size of S'};
TableProp{r, 2} = strcat(num2str(size(model.S,1)),'; ',num2str(size(model.S,2)));
r = r + 1;
```

Determine the rank of S.

```
TableProp(r, 1) = {'Rank of S'};
TableProp{r, 2} = strcat(num2str(rank(full(model.S))));
```

```
r = r + 1;
```

Determine the percentage of non-zero entries in the S matrix (nnz)

```
TableProp(r, 1) = {'Percentage nz'};
TableProp{r, 2} = strcat(num2str((nnz(model.S)/(size(model.S,1)*size(model.S,2)))));
r = r + 1;
```

View table.

```
TableProp
```

```
TableProp =
    'Model'                         []
    'Reactions'                 '7440'
    'Metabolites'               '5063'
    'Metabolites (unique)'      '2626'
    'Compartments (unique)'     '8'
    'Genes (unique)'            '1733'
    'Subsystems'                '100'
    'Deadends'                  '1332'
    'Size of S'                 '5063;7440'
    'Rank of S'                 '4666'
    'Percentage nz'             '0.0008373'
```

Determine blocked reactions properties (optional).

To evaluate the following model properties of bloack reactions, the solver package of IBM ILOG CPLEX is required. To install CPLEX refer to the Cobra Toolbox solver instalation guide, and change the solver to 'ibm_cplex' using the changeCobraSolver as shown above in equipment set-up.

- Determine the number of blocked reactions using fastFVA with 4 paralell workers (optional).

```
nworkers = 2;
solver = 'ibm_cplex';
setWorkerCount(nworkers);
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 2 workers.
Parallel computation initialized
```

```
tol = 1e-6;

TableProp(r, 1) = {'Blocked Reactions'};
[minFluxR3M, maxFluxR3M] = fastFVA(model, 0, 'max', solver);
```

```
 > The CPLEX version has been determined as 1271.

-- Warning:: You may only ouput 4, 7 or 9 variables.

 >> Solving Model.S. (uncoupled)
 >> The number of arguments is: input: 4, output 2.
 >> Size of stoichiometric matrix: (5063,7440)
 >> All reactions are solved (7440 reactions - 100%).
 >> 0 reactions out of 7440 are minimized (0.00%).
 >> 0 reactions out of 7440 are maximized (0.00%).
 >> 7440 reactions out of 7440 are minimized and maximized (100.00%).

 -- Starting to loop through the 2 workers. --

 -- The splitting strategy is 0. --

--------------------------------------------------------------------------------
--  Task Launched // TaskID: 2 / 2 (LoopID = 2) <> [3721, 7440] / [5063, 7440].
 >> Number of reactions given to the worker: 3720
 >> The number of reactions retrieved is 3720
 >> Log files will be stored at P:\Gitlab\fork-cobratoolbox\src\analysis\FVA\fastFVA\logFiles
 -- Start time:     Tue Jul 11 16:59:08 2017
 >> #Task.ID = 2; logfile: cplexint_logfile_2.log
        -- Minimization (iRound = 0). Number of reactions: 3720.
        -- Maximization (iRound = 1). Number of reactions: 3720.
 -- End time:     Tue Jul 11 17:05:21 2017
 >> Time spent in FVAc: 373.9 seconds.
--------------------------------------------------------------------------------
 ==> 50.0% done. Please wait ...

--------------------------------------------------------------------------------
--  Task Launched // TaskID: 1 / 2 (LoopID = 1) <> [1, 3720] / [5063, 7440].
 >> Number of reactions given to the worker: 3720
 >> The number of reactions retrieved is 3720
 >> Log files will be stored at P:\Gitlab\fork-cobratoolbox\src\analysis\FVA\fastFVA\logFiles
 -- Start time:     Tue Jul 11 16:59:08 2017
 >> #Task.ID = 1; logfile: cplexint_logfile_1.log
        -- Minimization (iRound = 0). Number of reactions: 3720.
        -- Maximization (iRound = 1). Number of reactions: 3720.
 -- End time:     Tue Jul 11 17:07:21 2017
 >> Time spent in FVAc: 499.0 seconds.
--------------------------------------------------------------------------------
 ==> 100% done. Analysis completed.
```

```
TableProp{r, 2} = num2str(length(intersect(find(abs(minFluxR3M) < tol), find(abs(maxFluxR3M) < tol))));
r = r + 1;
```

- Determine the percentage of blocked reactions.

```
TableProp(r, 1) = {'Blocked Reactions (Percentage)'};
TableProp{r, 2} = num2str(length(intersect(find(abs(minFluxR3M) < tol), find(abs(maxFluxR3M) < tol)))/length(model.rxns));
r = r + 1;
```

View table

```
TableProp
```

```
TableProp =
    'Model'                                   []
    'Reactions'                         '7440'
    'Metabolites'                       '5063'
    'Metabolites (unique)'              '2626'
    'Compartments (unique)'             '8'
    'Genes (unique)'                    '1733'
    'Subsystems'                        '100'
    'Deadends'                          '1332'
    'Size of S'                         '5063;7440'
    'Rank of S'                         '4666'
    'Percentage nz'                     '0.0008373'
    'Blocked Reactions'                 '2123'
    'Blocked Reactions (Percentage)'    '0.28535'
```

TIMING

This tutorial takes a few minutes depending on solver, computer, and model size. The most time consuming step is the flux variability analysis.

References

[1] Thiele et al., A community-driven global reconstruction of human metabolism, Nat Biotech, 2013.