Convert a reconstruction into a flux balance analysis model

Author: Ronan Fleming, Ines Thiele, University of Luxembourg

Reviewers:

INTRODUCTION

Even with quality control during the reconstruction process, it is not appropriate to assume that any reconstruction can be converted directly into a model and used to make predictions. A model must satisfy certain assumptions before it can be used to make reliable predictions. Depending on the type of model model, these assumptions will be different. Each assumption should be chemically or biologically motivated and expressed in an unambiguous manner and preferably both intuitively and mathematically. Flux balance analysis is a mathematical method widely used for studying genome-scale biochemical network. Here one aims to predict steady-state reaction fluxes, where there is a balance between production and consumption of each molecular species that is not exchanged across the specified boundary of a system. In this situation, one might obtain erroneous predictions if the system boundary is incorrectly specified. If a reconstruction contains one or more supposedly mass balanced reactions, but which are actually not mass balanced, such reactions in a model can lead to inadvertent leakage of a metabolite from the model, in violation of mass balance. Similarly, when generating a model for flux balance analysis, it is important to ensure that the network is flux consistent, that is, each reaction can carry a non-zero steady state flux.

Given a reconstruction with $\widehat{m}$ reactants involved in $\widehat{n}$ reactions, this tutorial demonstrates a method to identify and extract the largest subset of the reconstruction whose internal reactions are both stoichiometrically and flux consistent and whose external reactions are flux consistent. This model is then mathematically consistent with the basic requirements for generation of predictions using flux balance analysis. The identification of the component of the reconstruction that does not satisfy the aforementioned modelling conditions is also useful for targeting reconstruction effort towards resolving stoichiometric inconsistency or resolving flux inconsistency. The example used in this tutorial illustrates the process of extracting a model consistent with flux balance analsis, from a ReconX reconstruction.

PROCEDURE

Select reconstruction to convert into a model and enter parameters

Load the ReconX reconstruction, and save the original reconstruction in the workspace, unless it is already loaded into the workspace.

```
clear model
if ~exist('modelOrig','var')
    %select your own model, or use Recon2.0model instead
    if 1
        filename='Recon3D_301.mat'
        load(filename);
        model=Recon3D;
    else
        filename='Recon2.0model.mat';
        if exist('Recon2.0model.mat','file')==2
            model = readCbModel(filename);
        end
    end
    model.csense(1:size(model.S,1),1)='E';
    modelOrig = model;
else
    model=modelOrig;
end
```

Set the level of printing, zero for silent, higher for more output.

```
printLevel=2;
```

Choose the directory to place the results

```
basePath='~/work/sbgCloud/';
%resultsPath=[basePath '/programReconstruction/projects/recon2models/results/reconXs/'
resultsPath=[basePath '/courses/2019_Leiden_COBRA/practicalsDemo/Day4/' model.modelID];
resultsFileName=[resultsPath filesep model.modelID];
```

Create and enter the folder for the results if it does not already exist

```
if ~exist(resultsPath,'dir')
    mkdir(resultsPath)
end
cd(resultsPath)
```

Optionally create a diary to save the output in case it is very long, this makes it easier to search, especially when debugging the process during the early stages.

```
if 0
    diary([resultsFileName '_diary.txt'])
end
```

Overview some of the key properties of the reconstruction

Noting the initial size of the reconstruction is useful for comparisons later with subsets derived according to mathematical specifications.

```
[nMet,nRxn]=size(model.S);
fprintf('%6s\t%6s\n','#mets','#rxns')
```

```
 #mets   #rxns
```

```
fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' totals.')
```

```
  8399   13290    totals.
```

Make sure the stoichiometric matrix is stored in a sparse format as this accelerates computations with large networks

```
model.S=sparse(model.S);
```

Check in case the reconstruction is a model that is already ready for flux balance analysis

There is no need to run this live script any further if the reconstruction already satisfies the conditions necessary for flux balance analysis. That is if all internal reactants and reactions are stoichiometrically consistent, and all reactions are flux consistent, then the reconstruction satisfies the criteria to designate it a model ready for flux balance analysis.

SIntMetBool	m x 1 Boolean of metabolites heuristically though to be involved in mass balanced reactions.
SIntRxnBool	n x 1 Boolean of reactions heuristically though to be mass balanced.
SConsistentMetBool	m x 1 Boolean vector indicating consistent mets
SConsistentRxnBool	n x 1 Boolean vector indicating consistent rxns
fluxConsistentMetBool	m x 1 Boolean vector indicating flux consistent mets
fluxConsistentRxnBool	n x 1 Boolean vector indicating flux consistent rxns

```
if all(isfield(model,{'SIntMetBool','SIntRxnBool','SConsistentMetBool',...
        'SConsistentRxnBool','fluxConsistentMetBool','fluxConsistentRxnBool'}))
    if all(model.SIntMetBool & model.SConsistentMetBool)...
            && nnz(model.SIntRxnBool & model.SConsistentRxnBool)==nnz(model.SIntRxnBool
            && all(model.fluxConsistentMetBool)...
            && all(model.fluxConsistentRxnBool)
        fullyStoichAndFluxConsistent=1;
        fprintf('%s\n','Reconstruction is a model that is already ready for flux balanc
    end
    return
else
    fullyStoichAndFluxConsistent=0;
    fprintf('%s\n','Reconstruction must be tested to check if it is ready for flux bala
end
```

```
Reconstruction must be tested to check if it is ready for flux balance analysis
```

Manually remove certain reactions from the reconstruction

Before attempting to algorithmically remove stoichiometrically or flux inconsistent supposed internal reactions from a reconstruction to generate a model, there is an option to review the content of the reconstruction and manually identify reactions for removal. That is, there are two options:

A. Skip manual review of reconstruction content. Move to the next step.

B. Review the content of the reconstruction and omit any reactions that are assumed to be stoichiometrically or flux inconsistent. With respect to stoichiometric inconsistency, such reactions may be obviously mass imbalanced and not satisfy the heuristic conditions for indentification as an exernal reaction. Alternatively, such reactions may be identified by a previous pass through of this tutorial as being of unknown stoichometric consistent (model.unknownSConsistencyRxnBool(j)==1), after the largest stoichiometrically consistent subset of the network has been is identified. This is an iterative process where multiple rounds of identification of the largest stoichiometrically consistent set and manual curation of the remainder that is of unknown stoichiometric consistency is necessary.

```
if strcmp(filename,'Recon3.0model')
    modelOrig=model;
    if 0
        if 1
            %Rename some of the biomass reactions to make them more obviously exchange
            %reactions
            model.rxns{strcmp(model.rxns,'biomass_reaction')}= 'EX_biomass_reaction';
            model.rxns{strcmp(model.rxns,'biomass_maintenance')}= 'EX_biomass_maintenar
            model.rxns{strcmp(model.rxns,'biomass_maintenance_noTrTr')}= 'EX_biomass_ma

            %ATP hydrolysis is not imbalanced like all the other demand reactions so
            %give it a different accronym ATPM = ATP Maintenance
            bool=strcmp('DM_atp_c_',model.rxns);
            model.rxns{bool}='ATPM';
        end
        [model,removeMetBool,removeRxnBool] = manuallyAdaptRecon3(model,printLevel);
    else
        [model,removeMetBool,removeRxnBool] = manuallyAdaptRecon3Ines(model,printLevel)
    end
    [nMet0,nRxn0]=size(modelOrig.S);
    [nMet,nRxn]=size(model.S);
    if nMet0==nMet && nRxn0==nRxn && printLevel>0
        fprintf('%s\n','--- Manually removing rows and columns of the stoichiometric ma
        fprintf('%6s\t%6s\n','#mets','#rxns')
        fprintf('%6u\t%6u\t%s\n',nMet0,nRxn0,' totals.')
        fprintf('%6u\t%6u\t%s\n',nMet0-nMet,nRxn0-nRxn,' manually removed.')
        fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' remaining.')
    end
end
```

Remove any trivial rows and columns of the stoichiometric matrix

Remove any zero rows or columns of the stoichiometric matrix

```
modelOrig=model;
model=removeTrivialStoichiometry(model);
[nMet0,nRxn0]=size(modelOrig.S);
[nMet,nRxn]=size(model.S);
if nMet0==nMet && nRxn0==nRxn && printLevel>0
    fprintf('%s\n','---Checking for Remove any trivial rows and columns of the stoichic
    fprintf('%6s\t%6s\n','#mets','#rxns')
    fprintf('%6u\t%6u\t%s\n',nMet0,nRxn0,' totals.')
    fprintf('%6u\t%6u\t%s\n',nMet0-nMet,nRxn0-nRxn,' duplicates removed.')
    fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' remaining.')
end
```

```
---Checking for Remove any trivial rows and columns of the stoichiometric matrix----
 #mets     #rxns
  8399     13290      totals.
     0         0      duplicates removed.
  8399     13290      remaining.
```

Check for duplicate columns by detecting the columns of the S matrix that are identical upto scalar multiplication.

```
modelOrig=model;
dupDetectMethod='FR';
dupDetectMethod='S';
removeFlag=0;
[modelOut,removedRxnInd, keptRxnInd] = checkDuplicateRxn(model,dupDetectMethod,removeF
```

Remove any duplicate reactions, and uniquely involved reactants, from the stoichiometric matrix.

```
if length(removedRxnInd)>0
    irrevFlag=0;
    metFlag=1;
    %set all reactions reversible that are duplicates
    model.lb(removedRxnInd)=-model.ub(removedRxnInd);
    %remove duplicates
    model = removeRxns(model,model.rxns(removedRxnInd),irrevFlag,metFlag);
end
```

Display the statistics on the duplicate reactions,

```
[nMet0,nRxn0]=size(modelOrig.S);
[nMet,nRxn]=size(model.S);
if nMet0==nMet && nRxn0==nRxn && printLevel>0
    fprintf('%s\n','---Remove any duplicate reactions----')
    [nMet0,nRxn0]=size(modelOrig.S);
    [nMet,nRxn]=size(model.S);
    fprintf('%6s\t%6s\n','#mets','#rxns')
    fprintf('%6u\t%6u\t%s\n',nMet0,nRxn0,' totals.')
    fprintf('%6u\t%6u\t%s\n',nMet0-nMet,nRxn0-nRxn,' duplicates removed.')
    fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' remaining.')
end
```

```
---Remove any duplicate reactions----
 #mets   #rxns
  8399   13290    totals.
     0       0    duplicates removed.
  8399   13290    remaining.
```

Remove any duplicate reactions upto protons

Remove reactions reactions that differ only in the number of protons involved as substrates or products. Also remove exclusively involved reactants.

Save a temporary model for testing, before making any changes.

```
modelH=model;
```

Find the proton indicies in different compartments. A proton, with index i, is asumed to be represented by an abbreviation within model.mets{i} like h[*], where * denotes the compartment symbol.

```
nMetChars=zeros(length(modelH.mets),1);
for m=1:length(modelH.mets)
    nMetChars(m,1)=length(modelH.mets{m});
end
protonMetBool=strncmp(modelH.mets,'h',1) & nMetChars==length('h[*]');
if printLevel>2
    disp(modelH.mets(protonMetBool))
end
```

Zero out the proton stoichiometric coefficients from the temporary model for testing

```
modelH.S(protonMetBool,:)=0;
```

Check for duplicate columns, upto protons, by detecting the columns of the S matrix that are identical upto scalar multiplication.

```
dupDetectMethod='FR';
removeFlag=0;
[modelOut,removedRxnInd, keptRxnInd] = checkDuplicateRxn(modelH,dupDetectMethod,removeF
```

```
Checking for reaction duplicates by stoichiometry (up to orientation) ...
     Keep:     BTNt2    btn[e]     <=>    btn[c]
Duplicate:     BTNt4i    btn[e]     ->    btn[c]
     Keep:     GLCt1r    glc_D[e]     <=>    glc_D[c]
Duplicate:     GLCt2_2    glc_D[e]     <=>    glc_D[c]
Warning: Htg has more than one replicate
     Keep:     Htg        <=>
Duplicate:     HMR_1095        <=>
     Keep:     NACUP    nac[e]     ->    nac[c]
Duplicate:     NACDe    nac[c]     ->    nac[e]
     Keep:     ORNt4m    orn[m] + citr_L[c]     <=>    orn[c] + citr_L[m]
Duplicate:     r0947    orn[m] + citr_L[c]     ->    orn[c] + citr_L[m]
     Keep:     THYMDt1    thymd[e]     ->    thymd[c]
Duplicate:     THYMDtr2    thymd[c]     <=>    thymd[e]
     Keep:     VITD3tm    vitd3[m]     ->    vitd3[c]
Duplicate:     VITD3tm3    vitd3[c]     ->    vitd3[m]
```

Remove any duplicate reactions from the stoichiometric matrix, but do not remove the protons.

```
if length(removedRxnInd)>0
    irrevFlag=0;
    metFlag=0;%dont remove the protons
    model = removeRxns(model,model.rxns(removedRxnInd),irrevFlag,metFlag);
end
```

Display statistics of the removed reactions

```
if printLevel>0
    [nMet0,nRxn0]=size(modelOrig.S);
    [nMet,nRxn]=size(model.S);
    fprintf('%6s\t%6s\n','#mets','#rxns')
    fprintf('%6u\t%6u\t%s\n',nMet0,nRxn0,' totals.')
    fprintf('%6u\t%6u\t%s\n',nMet0-nMet,nRxn0-nRxn,' duplicate reactions upto protons r
    fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' remaining.')
end
```

```
  #mets     #rxns
   8399     13290     totals.
      0         7     duplicate reactions upto protons removed.
   8399     13283     remaining.
```

```
%model size
[nMet,nRxn]=size(model.S);
```

Heuristically identify exchange reactions and metabolites exclusively involved in exchange reactions

An external reaction is one that is heuristically identified by a single stoichiometric coefficient in the corresponding column of S, or an (abbreviated) reaction name matching a pattern (e.g. prefix EX_) or an external subsystem assignment. Any remaining reaction is assumed to be an internal reaction. If a reaction is not external then it is denoted an internal reaction. External reactants are exclusively involved in exchange reactions, and internal reactants otherwise. The findSExRxnInd function finds the external reactions in the model which export or import mass from or to the model, e.g. Exchange reactions, Demand reactions, Sink reactions.

```
if ~isfield(model,'SIntMetBool')  ||  ~isfield(model,'SIntRxnBool')
     model = findSExRxnInd(model,[],printLevel-1);
end
```

EXPECTED RESULTS

In the returned model, model.SIntRxnBool, is a boolean of reactions heuristically though to be mass balanced, while model.SIntMetBool is a boolean of metabolites heuristically though to be involved in mass balanced reactions.

CAUTION

The aforementioned assignments of external and internal reactions and reactants is the result of a heuristic and might result in one or more errors, either due to misspecification or because the names of external reactions and external subsystems often vary between laboratories.

Find the reactions that are flux inconsistent

Ultimately we seek to identify the set of stoichiometrically consistent reactions that are also flux consistent, with no bounds on reaction rates. However, finiding the stoichiometrically consistent subset can be demanding for large models so first we identify the subset of reactions that are flux consistent and focus on them.

```
modelOrig=model;
model.lb(~model.SIntRxnBool)=-1000;
model.ub(~model.SIntRxnBool)= 1000;
if 1
    if ~isfield(model,'fluxConsistentMetBool') || ~isfield(model,'fluxConsistentRxnBool
        param.epsilon=1e-4;
        param.modeFlag=0;
        param.method='null_fastcc';
        %param.method='fastcc';
        [fluxConsistentMetBool,fluxConsistentRxnBool,...
```

```
            fluxInConsistentMetBool,fluxInConsistentRxnBool,model]...
            = findFluxConsistentSubset(model,param,printLevel);
    end
    % Remove reactions that are flux inconsistent
    if any(fluxInConsistentRxnBool)
        irrevFlag=0;
        metFlag=1;
        model = removeRxns(model,model.rxns(fluxInConsistentRxnBool),irrevFlag,metFlag
        [nMet0,nRxn0]=size(modelOrig.S);
        [nMet,nRxn]=size(model.S);

        if printLevel>0
            fprintf('%s\n','-------')
            fprintf('%6s\t%6s\n','#mets','#rxns')
            fprintf('%6u\t%6u\t%s\n',nMet0,nRxn0,' totals.')
            fprintf('%6u\t%6u\t%s\n',nMet0-nMet,nRxn0-nRxn,' flux inconsistent reactio
            fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' remaining.')
            fprintf('%s\n','-------')
            if printLevel>1
                for n=1:nRxn0
                    if fluxInConsistentRxnBool(n)
                        fprintf('%15s\t%-100s\n',modelOrig.rxns{n},modelOrig.rxnNames{
                    end
                end
            end
        end
        %revise model size
        [nMet,nRxn]=size(model.S);

        %Recompute
        %Heuristically identify exchange reactions and metabolites exclusively involve
        %finds the reactions in the model which export/import from the model
        %boundary i.e. mass unbalanced reactions
        %e.g. Exchange reactions
        %     Demand reactions
        %     Sink reactions

        model = findSExRxnInd(model,[],0);
        if printLevel>0
            fprintf('%s\n','------end------')
        end
    end
end
```

```
 12156    Total reactions
  5966    Reversible reactions.
  6190    Irreversible reactions.
 11488    Flux consistent reactions, without flipping.
   292    Flux inconsistent irreversible reactions, without flipping.
   376    Flux inconsistent reactions, without flipping.
 11783    Flux consistent reactions.
    81    Flux inconsistent reversible reactions left to flip.
 11785    Flux consistent reactions.
    79    Flux inconsistent reversible reactions left to flip.
 11787    Flux consistent reactions.
    77    Flux inconsistent reversible reactions left to flip.
```

```
11789     Flux consistent reactions.
   45     Flux inconsistent reversible reactions left to flip.
11791     Flux consistent reactions.
   35     Flux inconsistent reversible reactions left to flip.
```

Find mass leaks or siphons within the heuristically internal part, without using the bounds given by the model

```
if 1
    modelBoundsFlag=0;
    leakParams.epsilon=1e-4;
    leakParams.method='dc';
    leakParams.theta=0.5;
    [leakMetBool,leakRxnBool,siphonMetBool,siphonRxnBool,leakY,siphonY,statp,statn] =..
        findMassLeaksAndSiphons(model,model.SIntMetBool,model.SIntRxnBool,...
        modelBoundsFlag,leakParams,printLevel);
end
```

Find the maximal set of reactions that are stoichiometrically consistent

```
if ~isfield(model,'SConsistentMetBool') || ~isfield(model,'SConsistentRxnBool')
    if strcmp(model.modelID,'HMRdatabase2_00')
        massBalanceCheck=0;
    else
        massBalanceCheck=1;
    end
    if 1
        [SConsistentMetBool,SConsistentRxnBool,SInConsistentMetBool,SInConsistentRxnBoo
            =findStoichConsistentSubset(model,massBalanceCheck,printLevel);
    else
        %print out problematic reactions to file
        resultsFileName=[resultsPath filesep model.modelID];
        [SConsistentMetBool,SConsistentRxnBool,SInConsistentMetBool,SInConsistentRxnBoo
            =findStoichConsistentSubset(model,massBalanceCheck,printLevel,resultsFileNa
    end
end

rxnBool=model.SInConsistentRxnBool & model.SIntRxnBool;
if any(rxnBool)
    if printLevel>0
        fprintf('%s\n','Stoichiometrically inconsistent heuristically non-exchange reac
    end
    for n=1:nRxn
        if rxnBool(n)
            fprintf('%20s\t%50s\t%s\n',model.rxns{n},model.rxnNames{n},model.subSystems
        end
    end
    if printLevel>0
        fprintf('%s\n','---------------')
    end
end

rxnBool=model.unknownSConsistencyRxnBool & model.SIntRxnBool;
```

```
if any(rxnBool)
    if printLevel>0
        fprintf('%s\n','Unknown consistency heuristically non-exchange reactions:')
    end
    for n=1:nRxn
        if rxnBool(n)
            fprintf('%20s\t%50s\t%s\n',model.rxns{n},model.rxnNames{n},model.subSystems
        end
    end
    if printLevel>0
        fprintf('%s\n','--------------')
    end
end
```

Sanity check of stoichiometric and flux consistency of model with open external reactions

```
    if  all(model.SIntMetBool & model.SConsistentMetBool)...
            && nnz(model.SIntRxnBool & model.SConsistentRxnBool)==nnz(model.SIntRxnBool
            && all(model.fluxConsistentMetBool)...
            && all(model.fluxConsistentRxnBool)

        [nMet,nRxn]=size(model.S);
        if printLevel>1
            fprintf('%6s\t%6s\n','#mets','#rxns')
            fprintf('%6u\t%6u\t%s\n',nMet,nRxn,' totals.')
            fprintf('%6u\t%6u\t%s\n',nnz(~model.SIntMetBool),nnz(~model.SIntRxnBool),'
        end

        checksPassed=0;
        %Check that all heuristically non-exchange reactions are also stoichiometricall

        %exchange reactions
        model.EXRxnBool=strncmp('EX_', model.rxns, 3)==1;
        %demand reactions going out of model
        model.DMRxnBool=strncmp('DM_', model.rxns, 3)==1;
        %sink reactions going into or out of model
        model.SinkRxnBool=strncmp('sink_', model.rxns, 5)==1;
        %all heuristic non-exchanges, i.e., supposedly all external reactions
        bool=~(model.EXRxnBool | model.DMRxnBool | model.SinkRxnBool);
        if nnz(bool & model.SIntRxnBool & model.SConsistentRxnBool)==nnz(model.SConsist
            checksPassed=checksPassed+1;
            if printLevel>1
                fprintf('%6u\t%6u\t%s\n',nnz(model.SIntMetBool),nnz(model.SIntRxnBool),
            end
        end

        %Check for mass leaks or siphons in the stoichiometrically consistent part
        %There should be no leaks or siphons in the stiochiometrically consistent part
        modelBoundsFlag=0;
        leakParams.epsilon=1e-4;
        leakParams.eta = getCobraSolverParams('LP', 'feasTol')*100;
        leakParams.method='dc';
        [leakMetBool,leakRxnBool,siphonMetBool,siphonRxnBool,leakY,siphonY,statp,statn
            =findMassLeaksAndSiphons(model,model.SConsistentMetBool,model.SConsistentRx

        if nnz(leakMetBool)==0 && nnz(leakRxnBool)==0 && nnz(siphonMetBool)==0 && nnz(s
            checksPassed=checksPassed+1;
            if printLevel>1
                fprintf('%6u\t%6u\t%s\n',nnz(leakMetBool | siphonMetBool),nnz(leakRxnBo
            end
        end

        %Check that the maximal conservation vector is nonzero for each the
        %internal stoichiometric matrix
        maxCardinalityConsParams.epsilon=1e-4;%1/epsilon is the largest mass considered
        maxCardinalityConsParams.method = 'quasiConcave';%seems to work the best, but s
        maxCardinalityConsParams.theta = 0.5;
        maxCardinalityConsParams.eta=getCobraSolverParams('LP', 'feasTol')*100;
        [maxConservationMetBool,maxConservationRxnBool,solution]=maxCardinalityConserva

        if nnz(maxConservationMetBool)==size(model.S,1) && nnz(maxConservationRxnBool)=
            checksPassed=checksPassed+1;
            if printLevel>1
                fprintf('%6u\t%6u\t%s\n',nnz(maxConservationMetBool),nnz(maxConservatio
            end
        end

        %Check that each of the reactions in the model (with open external reactions) i
        modelOpen=model;
        modelOpen.lb(~model.SIntRxnBool)=-1000;
        modelOpen.ub(~model.SIntRxnBool)= 1000;
        param.epsilon=1e-4;
        param.modeFlag=0;
        param.method='null_fastcc';
        [fluxConsistentMetBool,fluxConsistentRxnBool,fluxInConsistentMetBool,fluxInCons

        if nnz(fluxConsistentMetBool)==size(model.S,1) && nnz(fluxConsistentRxnBool)==s
            checksPassed=checksPassed+1;
            if printLevel>1
                fprintf('%6u\t%6u\t%s\n',nnz(fluxConsistentMetBool),nnz(fluxConsistentR
            end
        end

        if checksPassed==4
            %save the model with open exchanges as the default generic
            %model
            model=modelOpen;
            if printLevel>0
                fprintf('%s\n','Open external reactions is stoichiometrically and flux
            end
        end
        save([resultsFileName '_consistent.mat'],'model')
    end
```

REFERENCES

Gevorgyan, A., Poolman, M. G., Fell D., Detection of stoichiometric inconsistencies in biomolecular models. Bioinformatics, 24(19):2245–51, 2008.

Fleming, R.M.T., et al., Cardinality optimisation in constraint-based modelling: Application to Recon 3D (submitted), 2017.

Brunk, E. et al. Recon 3D: A resource enabling a three-dimensional view of gene variation in human metabolism. (submitted) 2017.