Testing chemical and biochemical fidelity

Authors: Ronan Fleming, Ines Thiele, University of Luxembourg.

Reviewer:

Introduction

Once a context-specific model is generated, but before a it is used to make predictions of biological relevance, it should be subjected to a range of quantitative and qualitative chemical and biochemical fidelity tests. The stoichiometric consistency tests should not be necessary if one starts with a generic model where the internal reactions are all stoichiometrically consistent then a context-specific model extracted from it should also be stoichiometrically consistent. Beyond chemical fidelity, it is also very important to test biochemical fidelity. Such tests are very specific to the particular biological domain one is modelling. Here we focus on human metabolism and use the Recon3.0model with all external reactions closed.

PROCEDURE

Load a model

Load Recon3.0model, unless it is already loaded into the workspace:

```
clear %model
if ~exist('modelOrig','var')
    filename='Recon3.0model';
    directory='~/work/sbgCloud/programReconstruction/projects/recon2models/data/reconXComparis
    model = loadIdentifiedModel(filename,directory);
    model.csense(1:size(model.S,1),1)='E';
    modelOrig = model;
else
    model=modelOrig;
end
```

Display the size of the model

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

```
 #mets  #rxns
  5835  10600  totals in Recon3model
```

Set the threshold to classify flux into non-zero and zero flux:

```
threshold=1e-6;
```

Production of mthgxl from 12ppd-S

Add sink reactions for either end of the proposed pathway:

```
model=modelOrig;
```

```
model = addSinkReactions(model,{'12ppd-S[c]','mthgxl[c]'},[-100 -1; 0 100]);
```

```
Warning: Metabolite 12ppd-S[c] not in model - added to the model
sink_12ppd-S[c] 12ppd-S[c]  <=>
sink_mthgxl[c] mthgxl[c]  ->
```

Change the objective to maximise the sink reaction for mthgxl[c]

```
model = changeObjective(model,'sink_mthgxl[c]');
```

Test if it is possible to attain a nonzero objective, and if it is compute a sparse flux vector:

```
sol = optimizeCbModel(model,'max','zero')
```

```
sol = 
         full: []
          obj: []
        rcost: []
         dual: []
       solver: 'gurobi'
    algorithm: 'default'
         stat: 0
     origStat: 'INFEASIBLE'
         time: 0.0295
        basis: []
            f: 0
            x: []
```

Check to see if there is a non-zero flux through the objective

```
if sol.stat==1
    fprintf('%g%s\n',sol.v(model.c~=0),' flux through the sink_mthgxl[c] reaction')
end
```

Display the sparse flux solution, but only the non-zero fluxes, above a specified threshold.

```
if sol.stat==1
    for n=1:nRxn
        if abs(sol.v(n))>threshold
            formula=printRxnFormula(model, model.rxns{n}, 0);
            fprintf('%10g%15s\t%-60s\n',sol.v(n),model.rxns{n}, formula{1});
        end
    end
end
```

ANTICIPATED RESULTS

If FBAsol.stat==0, then it is infeasible for the model to produce ATP from water, as expected. If FBAsol.stat==1, then the supposedly closed model can produce ATP from water. This indicates that there are stoichiometrically inconsistent reactions in the network, which need to be identified. See the tutorial on conversion of a reconstruction into a flux balance model for instructions how to approach this issue.

Metabolic task: 4abut -> succ[m]

Add sink reactions for either end of the proposed pathway:

```
model=modelOrig;
model = addSinkReactions(model,{'gly[c]','co2[c]','nh4[c]'},[-100 -1; 0.1 100; 0.1 100]);
```

```
Warning: Reaction with the same name already exists in the model, updating the reaction
sink_gly[c] gly[c]  <=> 
sink_co2[c] co2[c]  -> 
sink_nh4[c] nh4[c]  -> 
```

Change the objective to maximise the sink reaction for nh4[c]

```
model = changeObjective(model,'sink_nh4[c]');
```

Test if it is possible to attain a nonzero objective, and if it is compute a sparse flux vector:

```
sol = optimizeCbModel(model,'max','zero')
```

```
sol = 
         full: [10602×1 double]
          obj: 100
        rcost: []
         dual: []
       solver: 'gurobi'
    algorithm: 'default'
         stat: 1
     origStat: 'OPTIMAL'
         time: 0.9773
        basis: [1×1 struct]
            x: [10602×1 double]
            f: 100
            y: []
            w: []
            v: [10602×1 double]
```

Check to see if there is a non-zero flux through the objective

```
if sol.stat==1
    fprintf('%g%s\n',sol.v(model.c~=0),' flux through the sink_nh4[c] reaction')
end
```

```
100 flux through the sink_nh4[c] reaction
```

Display the sparse flux solution, but only the non-zero fluxes, above a specified threshold.

```
if sol.stat==1
    for n=1:nRxn
        if abs(sol.v(n))>threshold
            formula=printRxnFormula(model, model.rxns{n}, 0);
            fprintf('%10g%15s\t%-60s\n',sol.v(n),model.rxns{n}, formula{1});
        end
    end
end
```

```
 0.0333333          GLUDC h[c] + glu_L[c]  -> co2[c] + 4abut[c]
  -0.12381          r1088 h[e] + cit[e]  <=> h[c] + cit[c]
 0.0333333          r1702 na1[e] + gln_L[e] + gly[c]  -> na1[c] + gln_L[c] + gly[e]
 0.0166667          r2008 gly[c] + arg_L[e]  -> gly[e] + arg_L[c]
 0.0166667        RE3052C cpppg3[c]  -> 6 h[c] + C05770[c]
```

```
    0.12381        CITt4_4 4 na1[e] + cit[e]  <=> 4 na1[c] + cit[c]
 -0.0166667      GLYGLYCNc h2o[c] + glygly[c]  <=> 2 gly[c]
  -0.528571     GLYSNAT5tc h[c] + na1[e] + gly[e]  <=> h[e] + na1[c] + gly[c]
  0.0166667EX_argglygly[e] argglygly[e]  <=>
 -0.0166667     ARGGLYGLYt h[e] + argglygly[e]  <=> h[c] + argglygly[c]
  0.0166667     ARGGLYGLYr arg_L[c] + glygly[c]  <=> h2o[c] + argglygly[c]
  0.0166667           HMBS h2o[c] + 4 ppbng[c]  -> 4 nh4[c] + hmbil[c]
  0.0166667          UPP3S hmbil[c]  -> h2o[c] + uppg3[c]
  0.0166667         UPPDC1 4 h[c] + uppg3[c]  -> 4 co2[c] + cpppg3[c]
   0.966667      EX_gly[e] gly[e]  <=>
  -0.388095         GLYt2r h[e] + gly[e]  <=> h[c] + gly[c]
 -0.0333333    EX_gln_L[e] gln_L[e]  <=>
 -0.0166667    EX_arg_L[e] arg_L[e]  <=>
      -99.9      EX_nh4[e] nh4[e]  <=>
       99.9          NH4tb nh4[e]  <=> nh4[c]
  0.0166667      C05770te3 C05770[c]  -> C05770[e]
  0.0166667   EX_C05770[e] C05770[e]  <=>
 -0.0666667        PPBNGte ppbng[c]  <=> ppbng[e]
 -0.0666667    EX_ppbng[e] ppbng[e]  <=>
  0.0333333       HMR_9802 h2o[c] + gln_L[c]  -> nh4[c] + glu_L[c]
         -1    sink_gly[c] gly[c]  <=>
  0.0333333    DM_4abut[c] 4abut[c]  ->
```

ANTICIPATED RESULTS

If FBAsol.stat==1 then it is feasible to produce mitochondrial succinate from 4-Aminobutanoate. If FBAsol.stat==0, then this metabolic function is infeasible. This is not anticipated and indicates that further gap filling is required (cf Gap Filling Tutorial).

Metabolic task: gly -> co2 and nh4 (via glycine cleavage system)

Add sink reactions for either end of the proposed pathway:

```
model=modelOrig;
model = addSinkReactions(model,{'4abut[c]','succ[m]'},[-100 -1; 0 100]);
```

```
sink_4abut[c] 4abut[c]  <=>
sink_succ[m] succ[m]  ->
```

Change the objective to maximise the sink reaction for nh4[c]

```
model = changeObjective(model,'sink_succ[m]');
```

Test if it is possible to attain a nonzero objective, and if it is compute a sparse flux vector:

```
sol = optimizeCbModel(model,'max','zero');
```

Check to see if there is a non-zero flux through the objective

```
if sol.stat==1
    fprintf('%g%s\n',sol.v(model.c~=0),' flux through the sink_succ[m] reaction')
end
```

```
100 flux through the sink_succ[m] reaction
```

Display the sparse flux solution, but only the non-zero fluxes, above a specified threshold.

```
if sol.stat==1
    for n=1:nRxn
        if abs(sol.v(n))>threshold
            formula=printRxnFormula(model, model.rxns{n}, 0);
            fprintf('%10g%15s\t%-60s\n',sol.v(n),model.rxns{n}, formula{1});
        end
    end
end
```

```
   6.81818          ADK1m atp[m] + amp[m]  <=> 2 adp[m]
  -4.54545      EX_utp[e] utp[e]  <=>
   22.7273          FUMtm pi[m] + fum[c]  <=> pi[c] + fum[m]
  0.826446           GGNG Tyr_ggn[c] + 8 udpg[c]  -> 8 h[c] + 8 udp[c] + ggn[c]
  0.826446         GLBRAN glygn1[c]  -> glygn2[c]
  0.826446         GLGNS1 3 udpg[c] + ggn[c]  -> 3 h[c] + 3 udp[c] + glygn1[c]
    3.0303        GLPASE1 3 pi[c] + glygn2[c]  -> 3 g1p[c] + dxtrn[c]
   4.54545         GTHRDt h2o[c] + atp[c] + gthrd[c]  -> h[c] + adp[c] + pi[c] + gthrd[m]
  -4.54545           MDHm nad[m] + mal_L[m]  <=> h[m] + nadh[m] + oaa[m]
   13.6364           MMMm mmcoa_R[m]  <=> succoa[m]
   13.6364        MMTSADm nad[m] + coa[m] + 2mop[m]  -> h[m] + nadh[m] + mmcoa_R[m]
   36.3636           O2tm o2[c]  <=> o2[m]
   15.9091           PPAm h2o[m] + ppi[m]  -> h[m] + 2 pi[m]
  -22.7273         SUCD1m fad[m] + succ[m]  <=> fadh2[m] + fum[m]
  -13.6364        SUCOASm coa[m] + atp[m] + succ[m]  <=> adp[m] + pi[m] + succoa[m]
  -4.54545          UMPK3 utp[c] + ump[c]  <=> 2 udp[c]
   40.9091          r0178 h2o[m] + nad[m] + sucsal[m]  <=> 2 h[m] + nadh[m] + succ[m]
   22.7273          r0179 h2o[m] + nadp[m] + sucsal[m]  -> 2 h[m] + nadph[m] + succ[m]
  -36.3636          r0616 nad[m] + 4hpro_LT[m]  <=> 2 h[m] + nadh[m] + 1p3h5c[m]
  -22.7273          r0638 nadp[m] + ddcacoa[m]  <=> h[m] + nadph[m] + dd2coa[m]
  -13.6364          r0643 h2o[m] + nad[m] + 2mop[m]  <=> 2 h[m] + nadh[m] + HC00900[m]
  -4.54545          r0885 pi[m] + gthrd[c]  <=> pi[c] + gthrd[m]
  -4.54545          r0892 utp[c]  <=> utp[e]
  -4.54545          r1156 uri[c] + dutp[c]  <=> h[c] + dudp[c] + ump[c]
   9.09091          PPItm ppi[c]  <=> ppi[m]
   18.1818          FUMtr fum[e]  <=> fum[c]
  -13.6364  EX_HC00900[e] HC00900[e]  <=>
  -13.6364      HC00900t4 pi[e] + HC00900[c]  <=> pi[c] + HC00900[e]
  -4.54545           OAAt h[c] + oaa[c]  <=> h[e] + oaa[e]
  -4.54545      EX_oaa[e] oaa[e]  <=>
   4.54545        MALOAtm oaa[c] + mal_L[m]  <=> mal_L[c] + oaa[m]
  -13.6364         MMALtm HC00900[m]  <=> HC00900[c]
 -0.826446sink_Tyr_ggn[c] Tyr_ggn[c]  <=>
  -2.20386 sink_glygn2[c] glygn2[c]  <=>
    3.0303         DXTRNt dxtrn[c]  <=> dxtrn[e]
    3.0303    EX_dxtrn[e] dxtrn[e]  <=>
  -18.1818      EX_fum[e] fum[e]  <=>
  -36.3636       EX_o2[e] o2[e]  <=>
   13.6364       EX_pi[e] pi[e]  <=>
   4.54545      EX_uri[e] uri[e]  <=>
```

```
    -4.54545             FUM h2o[c] + fum[c]  <=> mal_L[c]
     9.09091            GALU h[c] + g1p[c] + utp[c]  <=> ppi[c] + udpg[c]
    -4.54545           NDPK6 atp[c] + dudp[c]  <=> adp[c] + dutp[c]
     36.3636             O2t o2[e]  <=> o2[c]
    -4.54545          URIt2r h[e] + uri[e]  <=> h[c] + uri[c]
    -63.6364        SUCSALtm sucsal[m]  <=> sucsal[c]
    -63.6364        SUCSALte sucsal[c]  <=> sucsal[e]
    -63.6364    EX_sucsal[e] sucsal[e]  <=>
     22.7273        HMR_3135 fad[m] + ddcacoa[m]  -> fadh2[m] + dd2coa[m]
     6.81818        HMR_3966 h2o[m] + atp[m]  -> h[m] + amp[m] + ppi[m]
     36.3636        HMR_4783 o2[m] + h[m] + 4hpro_LT[m]  -> 2 h2o[m] + 1p3h5c[m]
           1     DM_4abut[c] 4abut[c]  ->
```

REFERENCES

[fleming_cardinality_nodate] Fleming, R.M.T., et al., Cardinality optimisation in constraint-based modelling: illustration with Recon 3D (submitted), 2017.

[sparsePaper] Le Thi, H.A., Pham Dinh, T., Le, H.M., and Vo, X.T. (2015). DC approximation approaches for sparse optimization. European Journal of Operational Research 244, 26–46.