Adding biological constraints to a flux balance model

Note: This tutorial is a draft and needs completion. Contributions welcome!

Authors: Diana C. El Assal and Ronan M.T. Fleming, Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxembourg.

Reviewers:

Anne Richelle, Systems Biology and Cell engineering, University of California San Diego

Almut Heinken, Molecular Systems Physiology Group, LCSB, University of Luxembourg

INTRODUCTION

A metabolic model can be converted into a condition-specific model based on the imposition of experimentally derived constraints. Constraints can be defined, for example, by imposing upper and lower flux bounds for each reaction. There are several types of constraints that can be imposed in a metabolic model and that represent specific intra- and extracellular conditions, such as biomass maintenance requirements, environmental constraints, or maximum enzyme capacities.

In general, biomass constraints [1] are added as part of a biomass reaction. In some instances, however, a cell-type (e.g. neurons) does not divide, but is only required to turn over its biomass components. This tutorial is particularly relevant for such cases. Turnover rates are commonly expressed as half-lives ($t_{1/2}$) and represent the time required for half of the biomass precursor to be replaced [2].

Using the experimental literature, metabolite $t_{1/2}$ were collected and converted into turnover rates (λ):

$$(1)\ \ \lambda = \frac{ln(2)}{t_{1/2}}$$

We consider a biochemical network of m molecular species and n biochemical reactions. The biochemical network is mathematically represented by a stoichiometric matrix $S \in \mathcal{Z}^{m\times n}$. After calculating λ, we integrate it into the steady-state equation:

$$(2)\ \ Sv = \frac{dx}{dt}$$

The steady-state flux vector v and the change in abundance over time ($\frac{dx}{dt}$) share the same units, with x being the abundance of the corresponding biomass precursor.

Equation (2) can thus be re-written:

$$(3)\ \ Sv = \frac{dx}{dt} = \lambda x$$

PROCEDURE

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: C:\gurobi650\win64\matlab
   - [*---] TOMLAB_PATH: C:\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  active        0      0      0      0      -
 dqqMinos      active        0      -      -      -      -
 glpk          active        1      1      -      -      -
 gurobi        active        1      1      1      1      -
 ibm_cplex     active        1      1      1      -      -
 matlab        active        1      -      -      -      1
 mosek         active        0      0      0      -      -
 pdco          active        1      -      1      -      -
 quadMinos     active        0      -      -      -      0
 tomlab_cplex  active        1      1      1      1      -
 qpng          passive       -      -      1      -      -
 tomlab_snopt  passive       -      -      -      -      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 ...
 > The COBRA Toolbox is up-to-date.
```

Setting the optimization solver

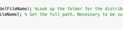
```
changeCobraSolver('gurobi','LP');
```

```
 > Gurobi interface added to MATLAB path.
```

Here, we use Recon2.0 model (distributed by the toolbox) for illustration, although any model can be used.

```
modelFileName = 'Recon2.0model.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);
modelOrig = model;
```

1. Environmental constraints



Environmental constraints are typically related to nutrient availability (e.g., glucose and oxygen). They can be defined using the function *changeRxnBounds* to set the minimal and maximal uptake and/or secretion rates possible in a specific condition. For example, in the caudate-putamen of the conscious rat, glucose consumption rate was found to range between -12.00 and -11.58 $\mu mol/gDW/hr$[3]. Therefore, the lower bound of the glucose exchange reaction (EX_glc(e)) can be set as follows:

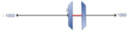
```
modelConstrained = model;
modelConstrained.c = 0*modelConstrained.c; % remove any objective function
```

```
modelConstrained = changeRxnBounds(modelConstrained, 'EX_glc(e)', −12, 'l');
```

Optionally, to further constrain the model, an upper bound can also be imposed to force the model to take up between 11.58 and 12 units of glucose

```
modelConstrained = changeRxnBounds(modelConstrained, 'EX_glc(e)', −11.58, 'u');
```

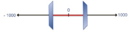
2. Internal enzymatic constraints



By convention, the bounds set on reaction rates in a metabolic model range from -1000 to 1000 and from 0 to 1000 for reversible and irreversible reactions, respectively [4]. Actually, the rate of a reaction is related to the activity of the enzyme catalyzing this reaction. Therefore, internal enzymatic constraints can be used to define the maximum capacity of a specified enzyme to catalyze a reaction (*Vmax*). For example, assuming that the reaction catalyzed by fructose-bisphosphate aldolase (FBA) has a *Vmax* of 128 units in our specific cell type, we can then add this constraint on the corresponding internal reaction FBA, as an upper bound.

```
modelConstrained = changeRxnBounds(modelConstrained, 'FBA', 128, 'u');
```

Optionally, if the reaction is reversible, the same constraint can be set as the lower bound, but with opposite signs.



```
modelConstrained = changeRxnBounds(modelConstrained, 'FBA', −128, 'l');
```

3. Constraints associated with biomass

In general, biomass constraints [1] are added as part of a biomass reaction by defining stoichiometric coefficients for each biomass precursor. For dividing cell types, the generic human biomass reaction available in Recon2 is formulated as follows:

```
printRxnFormula(modelConstrained, 'biomass_reaction');
```

```
biomass_reaction 20.6508 h2o[c] + 20.7045 atp[c] + 0.385872 glu_L[c] + 0.352607 asp_L[c] + 0.036117 gtp[c] + 0.279425 asn_L[c] + 0.505626 ala_L[c]
```

3.1 Biomass reaction

20.6508 h2o[c] + 20.7045 atp[c] + 0.38587 glu_L[c] + 0.35261 asp_L[c] + 0.036117 gtp[c] + 0.50563 ala_L[c] + 0.27942 asn_L[c] + 0.046571 cys_L[c] + 0.326 gln_L[c] + 0.53889 gly[c] + 0.39253 ser_L[c] + 0.31269 thr_L[c] + 0.59211 lys_L[c] + 0.35926 arg_L[c] + 0.15302 met_L[c] + 0.023315 pail_hs[c] + 0.039036 ctp[c] + 0.15446 pchol_hs[c] + 0.055374 pe_hs[c] + 0.020401 chsterol[c] + 0.002914 pglyc_hs[c] + 0.011658 clpn_hs[c] + 0.009898 dgtp[n] + 0.009442 dctp[n] + 0.013183 datp[n] + 0.053446 utp[c] + 0.013091 dttp[n] + 0.27519 g6p[c] + 0.12641 his_L[c] + 0.15967 tyr_L[c] + 0.28608 ile_L[c] + 0.54554 leu_L[c] + 0.013306 trp_L[c] + 0.25947 phe_L[c] + 0.41248 pro_L[c] + 0.005829 ps_hs[c] + 0.017486 sphmyln_hs[c] + 0.35261 val_L[c] -> 20.6508 h[c] + 20.6508 adp[c] + 20.6508 pi[c]

Any changes or adaptations can be introduced by adding a new formulation of the biomass function, using the function *addReaction*. For example, one can add the following new biomass reaction named *biomassReactionLipids*:

```
modelConstrained = addReaction(modelConstrained, 'biomasReactionLipids',  '20.6508 h2o[c] + 20.7045 atp[c] + 0.15446 pchol_hs[c] + 0.055374
```

```
test
biomasReactionLipids 20.6508 h2o[c] + 20.7045 atp[c] + 0.15446 pchol_hs[c] + 0.055374 pe_hs[c] + 0.020401 chsterol[c] + 0.011658 clpn_hs[c] + 0.00
```

4. Biomass maintenance constraints

To represent biomass maintenance (e.g. in neurons), the minimal biomass maintenance requirements can be used to set the corresponding constraints. Using the neurobiochemical literature, the degradation pathways for each biomass precursor has to be identified and the corresponding first reactions of these degradation pathways need to be mapped to ReconX. Using the fractional composition and the turnover rate of each biomass precursor, corresponding reaction rates ($\mu mol/gDW/hr$) are calculated as described above and in Table 1. These reaction rates represent the minimal requirements for biomass maintenance of neurons in the human grey matter and must therefore be imposed as a lower bound on the corresponding degradation reaction(s) of the different lipids, amino acids, and nucleic acids.

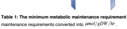
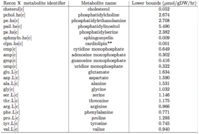


Table 1: The minimum metabolic maintenance requirement for neurons. This is a coarse-grained approximation of neuronal lipid, amino acid, and nucleic acid maintenance requirements converted into $\mu mol/gDW/hr$.



Recon X metabolite identifier	Metabolite name	Lower bounds (μmol/gDW/hr)
chsterol[c]	cholesterol	0.052
pchol_hs[c]	phosphatidylcholine	2.674
pe_hs[c]	phosphatidylethanolamine	2.708
pail_hs[c]	phosphatidylinositol	5.490
ps_hs[c]	phosphatidylserine	2.382
sphmyln_hs[c]	sphingomyelin	0.009
clpn_hs[c]	cardiolipin**	0.001
cmp[c]	cytidine monophosphate	0.649
amp[c]	adenosine monophosphate	0.302
gmp[c]	guanosine monophosphate	0.416
ump[c]	uridine monophosphate	0.322
glu_L[c]	glutamate	1.634
asp_L[c]	aspartate	1.590
ala_L[c]	alanine	1.531
gly[c]	glycine	1.032
ser_L[c]	serine	1.146
thr_L[c]	threonine	1.175
arg_L[c]	arginine	0.966
phe_L[c]	phenylalanine	0.771
pro_L[c]	proline	1.293
tyr_L[c]	tyrosine	0.745
val_L[c]	valine	0.940

**cardiolipin is also known as diphosphatidylglycerol

Calculation example of the minimal cholesterol maintenance requirement

i. Identify metabolite abundance

Assuming that the specific tissue type has a total dry weight lipid composition of 39.6%. This means that there is 0.396gLipid/gDWtissue. If cholesterol has a molar composition of 31.3%, then in total there is 0.124g cholesterol per gDW of tissue.

$$\frac{31.3 * 39.6 * 1gDW}{100 * 100} = 0.124gDW$$

```
abundance = (31.3*39.6*1)/(100*100);
```

ii. Calculate the molar abundance

In the experimental literature, cholesterol was also found to have a molar mass (M) of 386g/mol. Using equation (4), we can now convert the abundance (m) into molar units (n).

$$(4)\ n = \frac{m}{M}$$

```
M = 386; %g/mol
n = (abundance*1000000)/M; %micromol
```

iii. Calculate the corresponding flux value

Finally, we know that in the brain, cholesterol has a very slow turnover and a $t_{1/2}$ of 4320 hours. Using equation (3), we can now calculate the minimal cholesterol maintenance requirement in flux units (v_1).

```
halfLife = 4320;
turnover = log(2)/halfLife;
v1 = n * turnover
```

```
v1 = 0.0515
```

The minimal cholesterol maintenance requirement was calculated to be 0.0515$\mu mol/gDW/hr$ (Table 1). This value can now be used as a lower bound in the corresponding reaction.

4.1. Identification of degradation reactions for a biomass maintenance precursor

As previously mentioned, the degradation pathways for each biomass precursor can be identified using literature and the minimal maintenance requirements defined in section 4. can be used to constrain the first reactions of these degradation pathways. However, the identification of these reactions and the set-up of the associated constraints is not always straightforward. The following section presents the different common cases that can be encountered.

A. Single irreversible degradation reaction

In cases where only a single irreversible degradation reaction exists for a biomass maintenance precursor, the imposition of the constraint is straightforward. For example, the major cholesterol excretion pathway in the brain involves the hydroxylation of cholesterol into the oxysterol 24-hydroxycholesterol. Only a subset of neurons express this 24-hydroxylase enzyme (P45046A1r) and it is mainly found in dendrites and somata, rather than in axons or presynaptic terminals (reviewed in [5]).

Therefore, add a lower bound (v_1, calculated above) on P45046A1r (Table 1).

```
modelConstrained = changeRxnBounds(modelConstrained, 'P45046A1r', v1, 'l');
```

B. Single degradation reaction does not exist biochemically

A degradation reaction might not exist for a given biomass maintenance precursor. For example, the phospholipid cardiolipin is mainly present in the inner mitochondrial membrane, where it regulates the stability of the mitochondrial membrane protein complexes [6]. As part of mitochondria, cardiolipin reaches the lysosome during macroautophagy (reviewed in [7]). It is then degraded to form the negatively charged bis(monoacylglycero)phosphate (BMP) on internal membranes.

If the corresponding demand reaction does not exist in the model, a demand reaction can be added using the function *addDemandReaction*.

```
modelConstrained = addDemandReaction(modelConstrained, 'clpn_hs[c]');
```

```
DM_clpn_hs[c]  clpn_hs[c]  ->
```

Now, the constraint for cardiolipin (Table 1) can be imposed on the corresponding demand reaction

```
modelConstrained = changeRxnBounds(modelConstrained, 'DM_clpn_hs[c]', 0.001, 'l');
```

C. Single reversible degradation reaction

In cases where the biomass precursor degradation reaction is reversible, it first needs to be split into two irreversible reactions. To this end, define the set of reversible degradation reactions (sRxns) and convert them into two irreversible reactions (i.e., *sRXN_b* and *sRXN_f*, respectively backward and forward reactions) using *convertToIrreversible:*

i. Split the reversible reactions into irreversible

Select a set of reversible degradation reactions

```
sRxns = {'ASPTA'; 'GHMT2r'};
```

Copy the original model, except split a specific list of reversible degradation reactions into irreversible.

Split sRxns into irreversible reactions

```
[modelIrrev] = convertToIrreversible(modelConstrained,'sRxns',sRxns);
```

You can check if the conversion as been done properly by searching the split reactions

```
for j=1:length(sRxns)
    if isempty(findRxnIDs(modelIrrev, [sRxns{j} '_f']))
        error('Forward reaction not found')
    end
    if isempty(findRxnIDs(modelIrrev, [sRxns{j} '_b']))
        error('Reverse reaction not found')
    end
end
```

ii. Impose the calculated constraints

Examples are given for aspartate and glutamate (Table 1).

```
constraints = [1.590; 1.146];
```

You can also identify the new reaction names as follows:

```
rxns = setdiff(modelIrrev.rxns, modelConstrained.rxns)
```

```
rxns =
    'ASPTA_b'
    'ASPTA_f'
    'GHMT2r_b'
    'GHMT2r_f'
```

Using this list (rxns), manually identify the corresponding reactions that should be constrained

```
splitRxns= {'ASPTA_f'; 'GHMT2r_f'};
```

Identify the indices of these split reactions in the new model, using *findRxnIDs*

```
ind = findRxnIDs(modelIrrev, splitRxns);
```

Now, impose the constraints using a for loop. Note that you can easily account for experimental errors by defining a percentage error (e.g., *expError = 0.25*) for the constraint values.

```
expError = 0.25;
modelConstrained = modelIrrev;
for i = 1:length(splitRxns)
    modelConstrained = changeRxnBounds(modelConstrained, splitRxns{i,1},...
        constraints(i,1)-constraints(i,1)*expError, 'l');
end
```

D. Multiple irreversible degradation reactions

In some cases, several degradation pathways may be available for one biomass precursor. For example, in the brain, phosphatidylcholine (PC) can be degraded by 3 different metabolic pathways [8]:

- PCHOLP_hs: Phospholipase D acts on the choline/phosphate bond of PC to form choline and phosphatidic acid.
- PLA2_2: Phospholipase A2 acts on the bond between the fatty acid and the hydroxyl group of PC to form a fatty acid (e.g. arachidonic acid or docosahexaenoic acid) and lysophosphatidylcholine.
- SMS: Ceramide and PC can also be converted to sphingomyelin by sphingomyelin synthetase.

Define the set of potential reactions associated with the degradation of PC

```
multipleRxnList={'PCHOLP_hs', 'PLA2_2', 'SMS'};
```

Make sure that all the reactions are irreversible. The lower bounds should be 0 and the upper bounds 1000.

```
modelConstrained.lb(findRxnIDs(modelConstrained, multipleRxnList));
```

```
ans =

   0

   0
```

```
0
```

```
modelConstrained.ub(findRxnIDs(modelConstrained, multipleRxnList));
```

```
ans =

    1000

    1000

    1000
```

Constrain the weighted sum of fluxes to be above a lower bound (e.g. value of the maintenance requirement of PC in Table 1: d = 2.674 $\mu mol/gDW/hr$). The weight for each reaction are defined in *c.*

```
c=[1,1,1];
d=2.674;
ineqSense='G';
modelConstrainedAb=constrainRxnListAboveBound(modelConstrained,multipleRxnList,c,d,ineqSense);
rxnInd = findRxnIDs(modelConstrainedAb, multipleRxnList);
```

Check the constraints are there

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

```
modelConstrainedAb = 
                        S: [5063×7444 double]
                     rxns: {7444×1 cell}
                       lb: [7444×1 double]
                       ub: [7444×1 double]
                      rev: [7440×1 double]
                        c: [7444×1 double]
               rxnGeneMat: [7444×2194 double]
                    rules: {7444×1 cell}
                    genes: {2194×1 cell}
                  grRules: {7444×1 cell}
               subSystems: {7444×1 cell}
                 rxnNames: {7444×1 cell}
                rxnKeggID: {7440×1 cell}
     rxnConfidenceEcoIDA: {7440×1 cell}
    rxnConfidenceScores: {7440×1 cell}
              rxnsboTerm: {7440×1 cell}
           rxnReferences: {7440×1 cell}
            rxnECNumbers: {7440×1 cell}
                rxnNotes: {7440×1 cell}
                    mets: {5063×1 cell}
                        b: [5063×1 double]
                 metNames: {5063×1 cell}
              metFormulas: {5063×1 cell}
                metCharge: [5063×1 double]
               metCHEBIID: {5063×1 cell}
                metKeggID: {5063×1 cell}
             metPubChemID: {5063×1 cell}
           metInchiString: {5063×1 cell}
           metHepatoNetID: {5063×1 cell}
                metEHMNID: {5063×1 cell}
              ExchRxnBool: [7440×1 logical]
                EXRxnBool: [7440×1 logical]
                DMRxnBool: [7440×1 logical]
              SinkRxnBool: [7440×1 logical]
              SIntRxnBool: [7440×1 logical]
                  metHMDB: {5063×1 cell}
                  modelID: 'Recon2.0model'
                   csense: [5064×1 char]
                    match: [7444×1 double]
          reversibleModel: 0
                        C: [1×7444 double]
                        d: 2.6740
```

Solve the FBA problem with added constraints C*v >= d , with or without objective function

```
%modelConstrainedAb = changeObjective(modelConstrainedAb, 'DM_atp_c_');
FBAsolution = optimizeCbModel(modelConstrainedAb,'max',1e-6);
```

Check the values of the added fluxes

```
FBAsolution.x(rxnInd)
```

```
ans =
    2.0508
    0.2334
    0.3898
```

Therefore, when you solve the FBA problem with this last constraint, the sum of flux values associated with these three reactions should be greater than the value of *d*

```
sum(c*FBAsolution.x(rxnInd))
```

```
ans = 2.6740
```

```
return;
```

CRITICAL STEP: Collection of data and conversion of experimental fluxes (Timing: 4-8 weeks)

The most time consuming step when imposing constraints is the collection of required information. Depending on the available experimental literature, it can take between 4-8 weeks to retrieve the biomass composition and the turnover rates of the different biomass precursors. It is crucial to correctly convert the obtained data into the corresponding fluxes. It is recommended to first define the flux unit you wish to use. A common unit used for prokaryotic models is micromol per gramDryWeight per hour ($\mu mol/gDW/hr$). However, in the experimental literature, a wide range of units is provided. Therefore, after each conversion, it is strongly recommended to double check the calculations to avoid modelling artifacts. Once all the constraints are available, it can take less than 5 minutes to impose the constraints on the corresponding reaction bounds, according to the information provided in this tutorial.

ANTICIPATED RESULTS

After imposing the above constraints, we can now test the likely outcome of an optimisation problem using a constraint-based model. For example, we can take advantage of sparseFBA to identify the minimal set of essential reactions required to fulfill a certain objective function (e.g. DM_atp_c_).

```
originalTest = model;
originalTest = changeObjective(originalTest , 'DM_atp_c_');
[vSparseOriginal, sparseRxnBoolOriginal, essentialRxnBoolOriginal]  = sparseFBA(originalTest);
```

Display the number of essential reactions thata re required to carry flux to fulfill the objective function:

```
cnt=0;
for n=1:length(originalTest.rxns)
    if essentialRxnBoolOriginal(n,1)==true
        cnt=cnt+1;
    end
end
 fprintf('%g%s\n',cnt,' reactions essential to fulfill the objective function DM_atp_c_')
```

In the absence of constraints, the minimal set of reactions required to maximise the objective function is 111 essential reactions.

```
constrainedTest = modelConstrainedAb;
constrainedTest = changeObjective(constrainedTest, 'DM_atp_c_');
[vSparseConstrained, sparseRxnBoolConstrained, essentialRxnBoolConstrained]  = sparseFBA(constrainedTest);
```

Display the number of essential reactions thata re required to carry flux to fulfill the objective function:

```
cnt=0;
for n=1:length(constrainedTest.rxns)
    if essentialRxnBoolConstrained(n,1)==true
        cnt=cnt+1;
    end
end
 fprintf('%g%s\n',cnt,' reactions essential to fulfill the objective function DM_atp_c_')
```

After the addition of constraints, the minimal set of reactions required is increased to 172 essential reactions. Therefore, in this example, it is useful to integrate cell-type specific constraints to further define the minimal set of essential reactions. In most cases, constraints also allow us to alter the feasible solution space to obtain fluxes that better agree with the known physiology of the cell type.

REFERENCES

1. Feist, A.M. and Palsson, B.Ø. The biomass objective function. Current Opinion in Microbiology. 13(3), 344–349 (2010).

2. Kuhar, M.J. On the use of protein turnover and half-lives. Neuropsychopharmacology. 34(5), 1172–1173 (2008).

3. Sokoloff, L. et al. The [14C]deoxyglucose method for the measurement of local cerebral glucose utilization: theory, procedure, and normal values in the conscious and anesthetized albino rat. J Neurochem. 28(5):897-916 (1977).

4. Thiele, I. and Palsson B.Ø. A protocol for generating a high-quality genome-scale metabolic reconstruction. Nat. Protocols. 5(1), 93–121(2010).

5. Zhang, J. and Liu, Q. Cholesterol metabolism and homeostasis in the brain. Protein Cell. 6(4), 254-64 (2015).

6. Martinez-Vicente, M. Neuronal mitophagy in neurodegenerative diseases. Front. Mol. Neurosci. 8, 10:64 (2017).

7. Schulze, H. et al. Principles of lysosomal membrane degradation: cellular topology and biochemistry of lysosomal lipid degradation. Biochim. Biophys. Acta. 1793(4), 674-83 (2009).

8. Lajtha, A. and Sylvester, V. Handbook of Neurochemistry and Molecular Neurobiology. Springer; 2008. Available here.