6. Supplementary Material

6.1. Numerical Linear algebra

6.1.1. Rank Computation

To compute the rank of a matrix, we employed the sparse LU factorisation package LUSOL (Gill et al., 1987, 2005). Given a sparse matrix $A \in R^{m \times n}$, LUSOL computes factorisations of the form

$$P_1 A P_2 = LDU, \tag{12}$$

where P_1 and P_2 are permutations, L is lower trapezoidal with unit diagonals, D is diagonal and nonsingular, U is upper trapezoidal with unit diagonals, and the rank of each factor L, D, U is $r \leq \min(m, n)$. This is LUSOL's estimate of rank(A).

The permutations are chosen to keep L and U sparse, subject to bounds on the off-diagonal elements of L and U. Threshold Partial Pivoting (Gill et al., 1987) requires $|L_{ij}| \leq \tau$ for some tolerance $\tau \in (1, 10]$, where $\tau = 2$ is a reasonable choice. Threshold Rook Pivoting (Gill et al., 1987) also requires $|U_{ij}| \leq \tau$ and is likely to be more reliable on general sparse A. We have observed that net stoichiometric matrices $S = R - F$ have a sharply defined rank that can be estimated reliably by Threshold Partial Pivoting on either S or S^T and this is cheaper than applying Threshold Rook Pivoting. The same is true for estimating the rank of $F||R$.

6.1.2. Identification of dependencies

To investigate the rationale for row rank-deficiency of $A := F||R$ derived directly from a reconstruction, we used Threshold Partial Pivoting to obtain the factorisation (12). The row permutation P_1 partitions the rows as

$$P_1 A = \begin{bmatrix} B \\ C \end{bmatrix},$$

where $B \in R^{r \times n}$, $C \in R^{(m-r) \times n}$, and rank(B) = r. By definition of rank, the over-determined linear system $B^T W = C^T$ is consistent (has a solution W). The nonzero entries of each column of W reveal the dependencies between rows of B and C. We obtained W by solving $\min \|B^T W - C^T\|$ using sparse QR factorisation (`W = B'\C';` in MATLAB). The column permutation P_2 further partitions A as

$$P_1 A P_2 = \begin{bmatrix} B_1 & B_2 \\ C_1 & C_2 \end{bmatrix},$$

where B_1 is $r \times r$. We could obtain W more efficiently by solving $B_1^T W = C_1^T$, where $B_1 = L_1 U_1$ is already factorised in terms of the first r rows and columns of L and U. The nonzero entries of W reveal the dependencies between rows of B_1 and C_1. As each row of B_1 and C_1 corresponds to a different molecular species, one can use W to investigate the biochemical rationale for dependency among rows of $F||R$ if dependency is observed.

6.2. Combinatorial dependence in exceptional models derived from genome-scale reconstructions

Combinatorial dependence among the rows of $F||R$ implies that $F||R$ is row rank deficient. However the reverse implication is not necessarily true. This is because linear dependence depends on the nonzero numerical values of elements in $F||R$ whereas combinatorial dependence depends only on the sparsity pattern of $F||R$ and not the numerical values in $F||R$. Nevertheless, it is of interest to check if an $F||R$ that is row rank deficient also contains combinatorially dependent rows.

Only 3 of the 29 reconstructions subjected to the four conditions in Section 3 resulted in a row rank-deficient $F||R$, with rank at most 3 lower than the number of rows. If rank($F||R$) is less than the number of rows of $F||R$ then numerical linear algebra (cf. Section 6.1.2) can be used to test for dependency between rows to identify possible reasons for the row rank deficiency. The three models with rank-deficient $F||R$ were from compartmentalised genome-scale models. In each of the three models, one could find at least one dependency between a dependent molecular species (one dependent row of $F||R$) and a set of independent molecular species within the same sub-cellular compartment (a set of linearly independent rows of $F||R$).

Each dependency in $F||R$ was due to the existence of two disjoint sets of molecular species, one having a cofactor moiety in common and one having a non-cofactor moiety in common, with one cofactor and one non-cofactor always present in reactant complexes within that sub-cellular compartment. That is, neither the cofactor moiety nor the non-cofactor moiety was synthesised or degraded within that sub-cellular compartment. This reflected one or more reactions that were missing from the original reconstruction that would either synthesise or degrade the moiety within that sub-cellular compartment. Such reactions would give $F||R$ full row rank, as inevitably there would be at least one reaction where the cofactor and non-cofactor moiety would not both be represented within a reaction complex at the same time. Table 1 illustrates a specific example of such a dependency within a model derived from a *Saccharomyces cerevisiae* reconstruction (iMM904). Alternatively, the dependency reflected the omission of a reaction to transport either the cofactor or non-cofactor moiety into that sub-cellular compartment. Such a reaction would typically not simultaneously involve both the cofactor moiety and the non-cofactor moiety rendering $F||R$ of full row rank.

Table 1: An example of a set of endoplasmic reticulum reactions within a genome-scale *Saccharomyces cerevisiae* reconstruction (iMM904), that results in a rank deficient $F\|R$ even when all molecular species are stoichiometrically consistent and all reactions are net flux consistent (given exchange reactions, assuming each reaction is reversible and omitting nontrivial rows). Besides the 6 reactions illustrated, Phytosphingosine, Sphinganine, Tetracosanoyl-CoA, Hexacosanoyl-CoA and Phosphate are not involved in any other reactions within the endoplasmic reticulum. Phytosphingosine and Sphinganine form one set reactions, while Tetracosanoyl-CoA, Hexacosanoyl-CoA and Phosphate form another set. These sets are disjoint as they do not share a molecular species in common. Observe that the support of both disjoint sets is identical, i.e., all reactions contain at least one member of both sets in the same reaction complex. This renders the corresponding rows of $F\|R$ row rank deficient. In fact, the rank of these 5 rows is 4, hence leading to row rank deficiency of $F\|R$. A more comprehensive model would have the 3-ketodihydrosphingosine reductase reaction (Phytosphingosine + NADPH $\rightleftharpoons$ 3-Dehydrosphinganine + NADP) that would result a full row rank $F\|R$.

R1 alkaline ceramidase (ceramide-1)
R2 alkaline ceramidase (ceramide-1)
R3 alkaline ceramidase (ceramide-2)
R4 alkaline ceramidase (ceramide-2)
R5 sphingoid base-phosphate phosphatase (sphinganine 1-phosphatase)
R6 sphingoid base-phosphate phosphatase (phytosphingosine 1-phosphate)
A1 CERASE124er
A2 CERASE126er
A3 CERASE224er
A4 CERASE226er
A5 SBPP1er
A6 SBPP1er

	Reaction Name	R1	R2	R3	R4	R5	R6
Metabolite species name	Abbreviation	A1	A2	A3	A4	A5	A6
Ceramide-1 (Sphinganine:n-C24:0)	cer1_24[r]	-1	0	0	0	0	0
Ceramide-1 (Sphinganine:n-C26:0)	cer1_26[r]	0	-1	0	0	0	0
Ceramide-2 (Phytosphingosine:n-C24:0)	cer2_24[r]	0	0	-1	0	0	0
Ceramide-2 (Phytosphingosine:n-C26:0)	cer2_26[r]	0	0	0	-1	0	0
Coenzyme A	coa[r]	-1	-1	-1	-1	0	0
Proton	h[r]	-1	-1	-1	-1	0	0
Water	h20[r]	0	0	0	0	-1	-1
Sphinganine 1-phosphate	sph1p[r]	0	0	0	0	-1	0
Phytosphingosine 1-phosphate	psph1p[r]	0	0	0	0	0	-1
Phytosphingosine	psphings[r]	0	0	**1**	**1**	0	**1**
Sphinganine	sphgn[r]	**1**	**1**	0	0	**1**	0
Tetracosanoyl-CoA	ttccoa[r]	**1**	0	**1**	0	0	0
Hexacosanoyl-CoA (n-C26:0CoA)	hexccoa[r]	0	**1**	0	**1**	0	0
Phosphate	pi[r]	0	0	0	0	**1**	**1**

6.3. Reproduction of numerical results

All of the reconstructions and code required to reproduce the numerical results referred to in this paper are publicly available within the COBRA toolbox (Schellenberger et al., 2011) via `https://github.com/opencobra/cobratoolbox`. The steps are as follows:

1. Install MATLAB version 8.4.0.150421 (R2014b) or above. Earlier versions of MATLAB may also suffice, but have not been tested for this purpose.
2. Install the latest version of The COBRA toolbox (more recent than December 1, 2015). From a unix command line, enter the command:
   git clone https://github.com/opencobra/cobratoolbox.git
3. Optionally install a 64-bit Unix implementation of LUSOL
   `http://stanford.edu/group/SOL/software/lusol/`.
   From a Unix command line, enter the command
   git clone `https://github.com/nwh/lusol.git`.
   Installation is optional as otherwise the sparse LU factorization provided in MATLAB is employed.
4. The folder `cobratoolbox/testing/testModels/modelCollectionFR` contains each of the reconstructions in COBRA Toolbox format (one MATLAB .mat file for each reconstruction). Each .mat file was derived from the original SBML file that was published with the respective papers or provided as published updates to the original SBML file, as detailed within the function
   `cobratoolbox/testing/testModels/modelCitations.m`.
5. All numerical results can be reproduced by calling the MATLAB function
   `cobratoolbox/papers/Fleming/FR_2015/checkRankFRdriver.m`.
   This driver file passes each reconstruction to `checkRankFR.m`, which generates the corresponding model as detailed in Section 2.4.4 and uses numerical linear algebra to check rank($F||R$), as described in Section 6.1.1.

Supplementary references

Gill, P. E., Murray, W., Saunders, M. A., Wright, M. H., Apr. 1987. Maintaining LU factors of a general sparse matrix. Linear Algebra and its Applications 88–89, 239–270.

Schellenberger, J., Que, R., Fleming, R. M. T., Thiele, I., Orth, J. D., Feist, A. M., Zielinski, D. C., Bordbar, A., Lewis, N. E., Rahmanian, S., Kang, J., Hyduk, D., Palsson, B. Ø., 2011. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. Nature Protocols 6 (9), 1290–1307.