frontiers in Microbiology

METHODS
published: 17 June 2016
doi: 10.3389/fmicb.2016.00907

From DNA to FBA: How to Build Your Own Genome-Scale Metabolic Model

Daniel A. Cuevas[1], Janaka Edirisinghe[2], Chris S. Henry[2], Ross Overbeek[3], Taylor G. O'Connell[4] and Robert A. Edwards[1,4,5,6]*

[1] Computational Science Research Center, San Diego State University, San Diego, CA, USA, [2] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA, [3] Fellowship for Interpretation of Genomes, Burr Ridge, IL, USA, [4] Biological and Medical Informatics Research Center, San Diego State University, San Diego, CA, USA, [5] Department of Computer Science, San Diego State University, San Diego, CA, USA, [6] Department of Biology, San Diego State University, San Diego, CA, USA

Microbiological studies are increasingly relying on *in silico* methods to perform exploration and rapid analysis of genomic data, and functional genomics studies are supplemented by the new perspectives that genome-scale metabolic models offer. A mathematical model consisting of a microbe's entire metabolic map can be rapidly determined from whole-genome sequencing and annotating the genomic material encoded in its DNA. Flux-balance analysis (FBA), a linear programming technique that uses metabolic models to predict the phenotypic responses imposed by environmental elements and factors, is the leading method to simulate and manipulate cellular growth *in silico*. However, the process of creating an accurate model to use in FBA consists of a series of steps involving a multitude of connections between bioinformatics databases, enzyme resources, and metabolic pathways. We present the methodology and procedure to obtain a metabolic model using PyFBA, an extensible Python-based open-source software package aimed to provide a platform where functional annotations are used to build metabolic models (http://linsalrob.github.io/PyFBA). Backed by the Model SEED biochemistry database, PyFBA contains methods to reconstruct a microbe's metabolic map, run FBA upon different media conditions, and gap-fill its metabolism. The extensibility of PyFBA facilitates novel techniques in creating accurate genome-scale metabolic models.

Keywords: metabolic modeling, metabolic reconstruction, *in silico* modeling, flux-balance analysis, model SEED, genome annotation

OPEN ACCESS

Edited by:
Jessy L. Labbé,
Oak Ridge National Laboratory, USA

Reviewed by:
Bor-Sen Chen,
National Tsing Hua University, Taiwan
Vincent J. Denef,
University of Michigan, USA

****Correspondence:***
Daniel A. Cuevas
dcuevas08@gmail.com

Specialty section:
This article was submitted to Systems Microbiology, a section of the journal Frontiers in Microbiology

Received: *29 January 2016*
Accepted: *27 May 2016*
Published: *17 June 2016*

Citation:
Cuevas DA, Edirisinghe J, Henry CS, Overbeek R, O'Connell TG and Edwards RA (2016) From DNA to FBA: How to Build Your Own Genome-Scale Metabolic Model. Front. Microbiol. 7:907. doi: 10.3389/fmicb.2016.00907

INTRODUCTION

Since the dawn of genomics, homology-based algorithms and annotation databases have been used to infer meaning from raw sequences (Overbeek et al., 2000, 2003; Aziz et al., 2008), and papers describing microbial genomes have summarized the number of metabolic genes and breakdowns of their potential capacity. However, that information was usually presented *in absentia* the biochemical network that it purports to describe. The metabolic summary of a genome was limited to a few tables of higher metabolic categories. Genome-scale metabolic networks have the potential to completely change our perspective of microbial genomics and of the meaning inferred from a genome sequence (Oberhardt et al., 2011; Plata et al., 2015; Yurkovich and Palsson, 2016). By placing the genome annotation in the context of how the biochemical components of the cell combine to consume substrates, produce energy, and grow, genome-scale models demonstrate the breadth of our understanding of an organism whose genome has been sequenced, while also highlighting the gaps in our knowledge that further study will complete.

Flux-balance analysis (FBA), described elsewhere in this special issue have become the *de facto* standard method for predicting the fluxes through the reactions in the metabolic network, and thereby asserting which biochemical reactions are complete in the organism. FBA is a constraint-based linear optimization approach to solving the flow of compounds through a metabolic network in order to predict cellular phenotypes (Palsson, 2000; Edwards et al., 2002; Orth et al., 2010). The reactions are written as equations, with compounds being converted from substrates to products. A single equation is included in the system that represents the *objective function*, the equation that is targeted to be optimized. In order to get growth (biomass production), ATP production, or any other output of the system, the system of equations representing the cell must produce a solution that results in flux through that single equation that represents the objective function; the optimization is typically to maximize the amount of flux through that equation. In modeling bacterial cells there are almost always more reactions than there are compounds (whose concentrations are unknown) that describe the system. For example, the *Citrobacter* model we recently published recently contains 1,399 reactions (columns) and 1,301 compounds (rows) (Cuevas et al., 2014). Therefore, these models are mathematically underdetermined and the only way to solve them is to apply specific constraints to the system (Kauffman et al., 2003).

The process of running FBA can be broken down into two broad objectives: creating the mathematical model and solving the mathematical model. Solving the mathematical model is straightforward and is usually performed by an optimization library. There are a number of alternatives including the Open Source Gnu Linear Programming Kit (GLPK) (Makhorin, 2008), the commercial (MATLAB, 2012) linprog[1], and IBM ILOG CPLEX Optimization Studio (IBM ILOG, 2014) is not the focus of this work. Creating the mathematical model is much more complex, as it requires incorporating biological knowledge to transition between DNA sequence, functional roles, enzymes, and reactions. Including other metabolic-related sources of information has also been used to build these models (Lee et al., 2006; Raman and Chandra, 2009; Carrera et al., 2014; Liu et al., 2014). There are several software packages designed to do some or all of these steps for you, such as the COBRA Toolbox (Schellenberger et al., 2011; Ebrahim et al., 2013), KBase (Overbeek et al., 2013), the Systems Biology Research Tool (Wright and Wagner, 2008), FASIMU (Hoppe et al., 2011), CellNetAnalyzer (Klamt and von Kamp, 2011), the Model SEED (DeJongh et al., 2007; Devoid et al., 2013), and others (Lakshmanan et al., 2012; Hamilton and Reed, 2014).

In this paper we describe the process of generating a metabolic reconstruction and running FBA starting with a genome sequence. We demonstrate how to identify the reactions present in a model derived from a genome, and how to convert those reactions to a stoichiometric matrix. We demonstrate how to identify additional reactions that need to be included in the model, and reactions that can be excluded, and how to test the model under different growth conditions. We introduce a new open source library, PyFBA, that allows bioinformaticians to build and explore FBA models using the Python programming language and that is freely available to all researchers. We explain each of the steps required to go from DNA to FBA for the bioinformatician.

[1]http://www.mathworks.com/help/optim/ug/linprog.html

FROM DNA TO FBA

The steps from DNA to FBA include identifying the functional roles in the genome; connecting those roles to enzyme complexes and then to reactions; converting those reactions to equations that describe the conversion of substrates to products; defining the growth media and external conditions; and testing growth of that model. Usually, developing a complete metabolic model requires several iterations of adding reactions to enable the model to grow and removing reactions to limit the growth of the model under conditions where it should not grow. We discuss each of these steps individually below.

PyFBA

We have developed a Python code base, PyFBA, that allows you to build a genome-scale metabolic model and run FBA on that model. The PyFBA code is available from GitHub or the Python Package Index repository under the MIT License (Cuevas et al., 2016a,b). PyFBA works with the GNU Linear Programming Kit (GLPK) or the IBM ILOG CPLEX Optimization Studio for solving the linear system. In the examples below we use this code to demonstrate how to go from DNA to FBA. To install the PyFBA code, see the detailed instructions available online at http://linsalrob.github.io/PyFBA/installation.html.

Genome Annotations

The first step in building a metabolic model of an organism is to identify all the genes present in that organism. There are a number of tools for genome annotation, including RAST (Aziz et al., 2008; Overbeek et al., 2013), PROKKA (Seemann, 2014), BG7 (Tobes et al., 2015), Blast2GO (Conesa et al., 2005), and BASys (Van Domselaar et al., 2005). Most of these tools take unannotated contigs, and iterate through steps for accurately identifying the protein- and RNA-encoding genes and assigning functional roles to those genes. Although all of these tools will identify most of the metabolic genes in the genome and will provide accurate annotations of those genes [including Enzyme Commission (EC) numbers, see below], connecting those annotations to enzymes and then to reactions is a complex undertaking for the output from most tools. In this paper we use annotations generated by RAST to connect to biochemical reactions encoded by the Model SEED as both the functional roles and connections to enzymes are publicly available and frequently updated. PyFBA does not require RAST annotations, but does require a connection from annotation to biochemistry.

The list of functional roles in a genome can be downloaded from the RAST website in several different formats: for example, from the *Job Details* page of an annotated genome, the annotations can be downloaded as spreadsheets (the easiest to use

in model building), GenBank files, GFF files, or RAST genome directories.

Converting Functional Roles to Reactions

After identifying the protein encoding genes present in the organism, and assigning functions to those proteins, the enzyme complexes that are created by those proteins must be characterized. EC numbers (Webb, 1992) are most often used when making these mappings between different repositories because they are the most widely applied annotation to gene products. However, EC numbers do not cover all reactions in microbial cells, and with different annotation naming conventions and nomenclature, automated processes to compile the list of reactions is extremely difficult. However, using a single database (e.g., the SEED that underlies the RAST platform) provides a convenient connection between functions, enzyme complexes, reactions, and compounds because of the consistent work of the annotators.

Enzyme complexes can be formed by one or several functional roles, and each functional role can be involved in one or more complexes, illustrating a many-to-many relationship (**Figure 1**). For example, the functional role "Phosphoenolpyruvate-protein phosphotransferase of PTS system (EC 2.7.3.9)" encoded by the *ptsI* gene in *Escherichia coli* is involved in several different complexes each associated with the import of a different sugar (**Figure 1A**); the Ubiquinol-cytochrome C complex requires ten different functional roles each encoded by a separate gene (**Figure 1B**); and the role "Alkaline phosphatase (EC 3.1.3.1)" encoded by the *phoA* gene in *E. coli* is in a complex by itself (**Figure 1C**). The first step in identifying the reactions that are encoded by a genome is therefore to convert the functional roles associated with the proteins identified in the genome to enzyme complexes that are functional in the cell. Comparably, each reaction in a cell can require one or more complexes, while each complex can be involved in one or more reactions, thus creating another many-to-many relationship. For example, the complex created by alkaline phosphatase is responsible for many dephosphorylation reactions (**Figure 1C**) while the dodecomeric glutamine synthetase catalyzes a single reaction (**Figure 1D**).

To convert functional roles to reactions roles must first be connected to enzyme complexes. If the annotation system used to identify the roles in the genome only provides EC numbers, these need to be connected to complexes. Most subunits of the same enzyme complex are given the same EC number; for example all the subunits of ATP synthase are given the EC number 3.6.3.14, which facilitates joining reactions into complexes. The connections between functional roles (and EC numbers in particular) and reactions or pathways can be obtained from several public resources, such as EXPASY[2] (Gasteiger et al., 2003), the KEGG REACTION database[3] (Kanehisa et al., 2004), MetaCyc[4] (Caspi et al., 2014), and BRENDA[5] (Schomburg et al., 2002). As an alternative, the Model SEED[6] (Overbeek et al., 2013) maintains a mapping between functional roles and complex IDs and a separate mapping between complex IDs and reactions.

Converting Reactions to a Stoichiometric Matrix

A genome-scale metabolic model starts with a list of reaction equations, compounds, and compartments, and for the mathematical solution we convert that to a *stoichiometric matrix,* essentially a table of reactions and compounds (**Figures 2A,B**). The stoichiometric matrix provides the first level of constraint on the metabolic system — it contains only those reactions and their associated metabolites present within the network, defining the feasible space of phenotypes the system can express. The cells of the matrix represent the relationship between each of the compounds and each of the reactions in the network; thus, a reaction that is not included in the stoichiometric matrix is not included in the model. All phenotypes recognized in the cell must be included in the stoichiometric matrix for an accurate metabolic model.

To construct the stoichiometric matrix, all of the compounds used in all the reactions are stored in the rows of a matrix. All the reactions used in the model are stored in the columns of the matrix, and the individual cells contain the stoichiometry of each compound in each reaction, with negative values indicating that the compound is consumed in the reaction; zero indicating that the compound is not involved in the reaction; and positive values indicating that the compound is produced by the reaction. Most of the values in the stoichiometric matrix are zero because most of the compounds are not involved in many reactions. The dimensions of the matrix are the number of compounds and the number of reactions; thus for the *Citrobacter* model discussed earlier (Cuevas et al., 2014) the matrix was 1,301 × 1,399, providing 1,820,099 possible combinations of reactions and compounds, but only 6,355 values (0.35%) in the matrix are nonzero. **Table 1** illustrates a stoichiometric matrix for the glycolysis pathway (9 reactions, 18 compounds) included in the *Citrobacter* model. The cells that contain a zero have been left blank for visual purposes.

Compounds in the stoichiometric matrix are also denoted by the compartment in which they are located to differentiate which compounds are required inside the cell from those required outside the cell. The location also provides a convenient mechanism to constrain the model based on which compounds are in the media and which can be transported (see below). Bacterial models typically use just two compartments: intra- or extra-cellular, while models of Eukaryotes often include other compartments, such as the mitochondria or chloroplast (Seaver et al., 2012). For the purpose of these models the Gram negative periplasmic space and things anchored to the outer cell wall are typically considered extracellular. For example, the Gram

[2]http://www.expasy.org/

[3]http://www.genome.jp/kegg/pathway.html

[4]http://metacyc.org/

[5]http://www.brenda-enzymes.info/

[6]http://www.theseed.org/models/

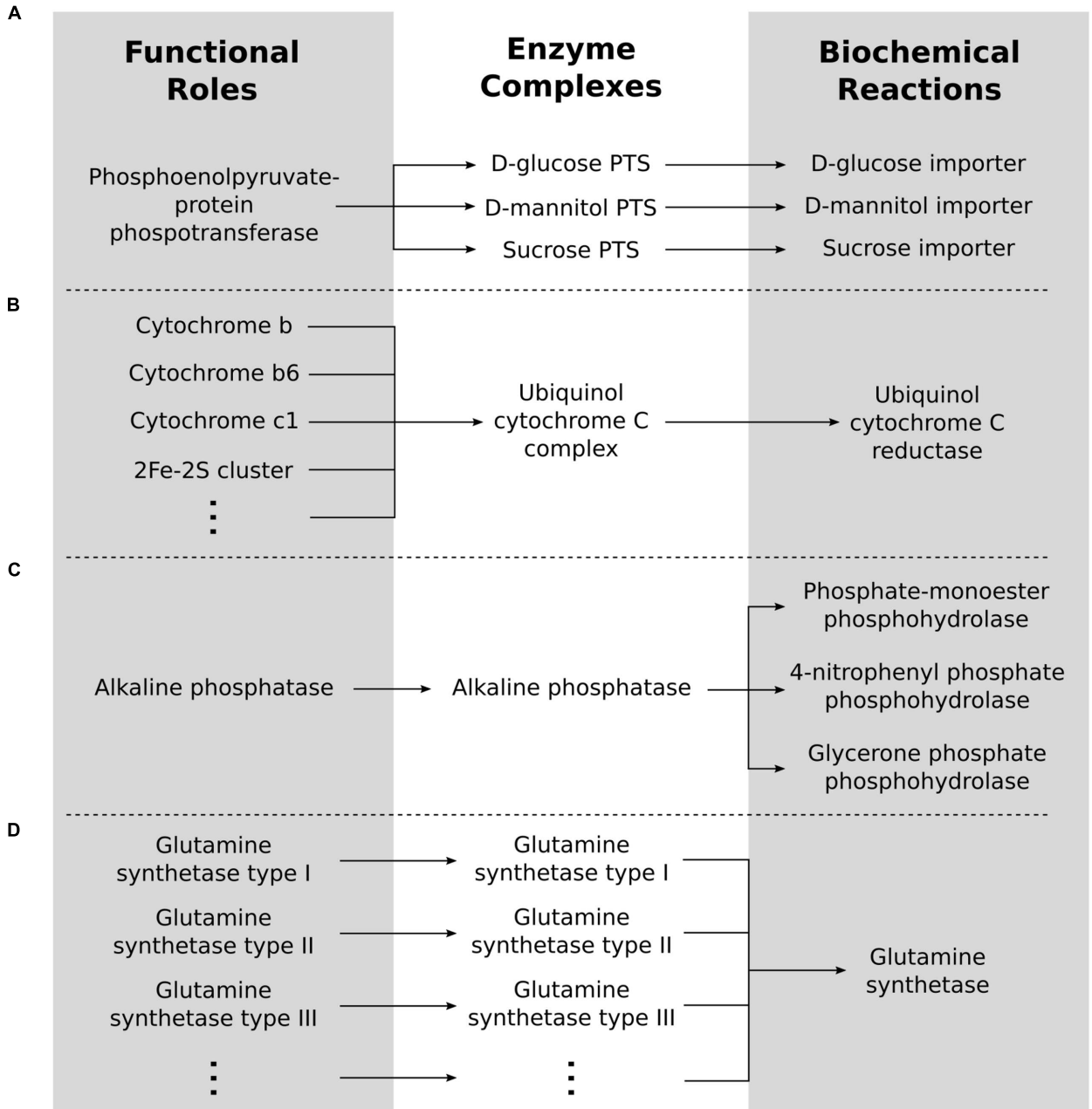


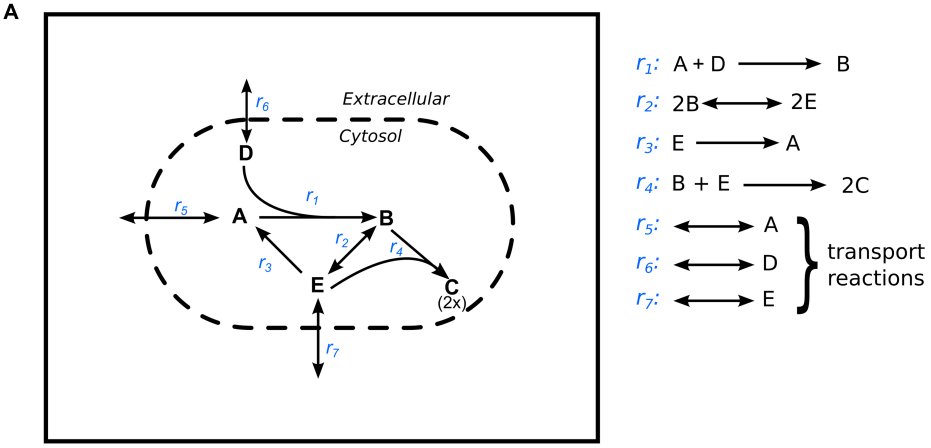
FIGURE 1 | Roles to complexes to reactions. Functional roles have a many-to-many relationship with enzyme complexes. Similarly, enzyme complexes have a many-to-many relationship with biochemical reactions. **(A)** A one-to-many relationship from roles to complexes. **(B)** A many-to-one relationship from roles to complexes. **(C)** A one-to-many relationship from complexes to reactions. **(D)** A many-to-one relationship from complexes to reactions.

negative *Citrobacter* model includes 175 compounds in the extracellular compartment.

Preparing the stoichiometric matrix requires iterating through all the reactions and all the compounds associated with those reactions, and creating a table. If the compounds are consumed the coefficient of that compound is negated in the stoichiometric matrix cell. If the reaction is reversed, the coefficients of the appropriate compounds may be negated (although this can also be controlled by the reaction bounds, described below). An example stoichiometric matrix is shown in **Figure 2** and provided in the supplementary table as a tab-separated text file (**Supplementary Material 1**) that can be imported into any office suite of programs.

Media Formulation

The media in which the organism is growing defines another constraint imposed upon the metabolic model. Encoding defined media is straightforward, and recipes for almost all media



B

$\boldsymbol{S}$ =

	r_1	r_2	r_3	r_4	r_5	r_6	r_7
A	-1	0	1	0	1	0	0
B	1	-2	0	-1	0	0	0
C	0	0	0	2	0	0	0
D	-1	0	0	0	0	1	0
E	0	2	-1	-1	0	0	1

(Stoichiometric values)

$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}$

(Metabolic flux values)

C

Objective function

$\max Z = v_4$

(Production of compound C)

Subject to

$$\boldsymbol{S}\vec{v} = \vec{0} = \begin{cases} \frac{dA}{dt} = -v_1 + v_3 + v_5 \\ \frac{dB}{dt} = v_1 - 2v_2 - v_4 \\ \frac{dC}{dt} = 2v_4 \\ \frac{dD}{dt} = -v_1 + v_6 \\ \frac{dE}{dt} = 2v_2 - v_3 - v_4 + v_7 \end{cases} \qquad \begin{array}{r} 0 \le v_1 < \infty \\ -\infty < v_2 < \infty \\ 0 \le v_3 < \infty \\ 0 \le v_4 < \infty \\ 0 \le v_5 \le \infty \\ -\infty < v_6 < \infty \\ 0 \le v_7 \le \infty \end{array}$$

(Steady state system) (Reaction bounds)

FIGURE 2 | Flux-balance analysis (FBA). (A) Example of a bacterial metabolic model displaying two compartments separated by a dashed boundary (extracellular and cytoplasm), seven reactions labeled in blue text (four intracellular and three transporters), and five compounds. **(B)** The stoichiometric matrix $\boldsymbol{S}$ with corresponding stoichiometric coefficients, and the flux vector $\boldsymbol{v}$. Each matrix-cell represents the number of compound molecules required for the particular reaction. The integer sign denotes the compound as a reactant (negative value) or as a product (positive value). A zero means the compound is not involved in the reaction. Reversible reactions are typically present in the matrix in one direction. In the instance that a reaction is reversed in the solution, the metabolic flux value for the corresponding reaction will be negative, thus indicating a switch in directionality. **(C)** The linear programming problem. The mass balance equations constrain the change of compound concentration over time to zero. The final constraints on the system are the physicochemical enzymatic bounds. These bounds signify directionality (e.g., v_3 has a lower bound of 0 and can only produce compound A, whereas v_2 is completely unbounded and can proceed in both directions). The objective function is indicated to maximize the flux of v_4 (i.e., the production of compound **C**).

[including complex media such as lysogeny broth (LB) media] are available online. The only difficulty is to ensure correlation between the compound names used in the media formulation and the compound names used in the metabolic reactions. One approach to overcome that obstacle, used, for example, by the Model SEED, is to use a separate database of compounds with unique IDs. By default, of course, the media components have an extracellular location, and the presence of transporters is

TABLE 1 | Glycolysis portion of stoichiometric matrix.

Reaction ID/Compound	Rxn 00216	00558	00545	00786	00781	01100	01106	00459	00148
ATP	−1		−1			1			1
ADP	1		1			−1			−1
D-Glucose	−1								
D-Glucose-6-Phosphate	1	−1							
D-Fructose-6-Phopshate		1	−1						
D-Fructose-1,6-Biphosphate			1	−1					
Glyceraldehyde-3-Phosphate				1	−1				
Glycerone-phosphate				1					
NAD					−1				
Phosphate					−1				
1,3-Bisphospho-D-Glycerate					1	−1			
NADH					1				
H+						1			
3-Phosphogylcerate						1	−1		
2-Phospho-D-Gylcerate							1	−1	
H2O								1	
Phosphoenolpyruvate								1	−1
Pyruvate									1

required to move them into the cell; another constraint imposed on the genome-scale metabolic model. Historically, identification of transporter proteins is challenging because they are largely homologous to each other and only differ by their substrate specificity (Marger and Saier, 1993; Saier, 1994). Latitude is therefore often given in the assertion of which transporter pathways a cell actually has, especially for some of the less well characterized biochemical compounds. Small molecules and ions may diffuse into and out of the cell as well as being actively transported across the membrane, and protein-free reactions invoking those diffusions are included in the cellular model.

Uptake and Secretion from the Media

The linear programming solution to the genome-scale metabolic model requires that compounds are taken up from the outside (i.e., the media components), and also that some compounds are secreted from the cell (waste products). Moreover, each compound in the stoichiometric matrix must be balanced for a solution to the problem. Therefore, a set of unconstrained reactions are generated that consume all external compounds in the model, but do not do anything with those compounds (in essence, they disappear from the equation; in practice this is akin to them being diluted to extinction in the growth media). These reactions also provide the media compounds as if from nowhere (in practice this is akin to the media compounds diffusing toward the cell as they are consumed). These reactions are sometimes called drain flux reactions or external reactions, and in PyFBA they are called uptake and secretion reactions. Since the set of uptake and secretion reactions is dependent on the external compounds produced by the model and the external compounds present in the media, it is generally calculated for each model upon creation.

Reaction Bounds

In solving linear equations of underdetermined systems, a defined solution space has to be provided that limits the parameters applied to each of the reactions. It is highly unlikely that one, or a few, reactions would have extremely high fluxes through them while the other reactions in the cell have moderate or low fluxes. Therefore, each reaction in the stoichiometric matrix is controlled by a set of *reaction bounds* that limit the flux through that reaction. In most cases, the reaction bound also assigns the directionality of the reaction. As shown in **Figure 2A**, the direction of reaction *r1* is set from left to right. This is controlled by setting the lower bound of reaction r_1 to zero (**Figure 2C**), thus preventing reaction r_1 from producing compounds A and D by the linear solver. Furthermore, an alternative approach to designating the directionality of a reaction is to limit the flux through the reaction by negating the coefficients in the equations. The limits are conventionally positive if a reaction proceeds from left to right; negative if a reaction proceeds from right to left; and positive and negative if a reaction is bidirectional. The reactions bounds are applied to the linear solver before the stoichiometric matrix is solved.

The media formulation (described above) is typically encoded as a list of compounds that are present in the media, and this is used to determine the reaction bounds of the reactions that transport these compounds into the cell. As we noted, there are a series of uptake and secretion reactions that mimic the diffusion of a media compound toward the cell, and the diffusion of a waste product away from the cell. By setting the bounds of these reactions appropriately, we can control the growth conditions of the cell. If the reaction bounds for diffusion of media components toward the cell are set to allow production of those components and if they are required for growth, they can be consumed. Similarly, the reaction bounds are set to only allow consumption of waste products so that

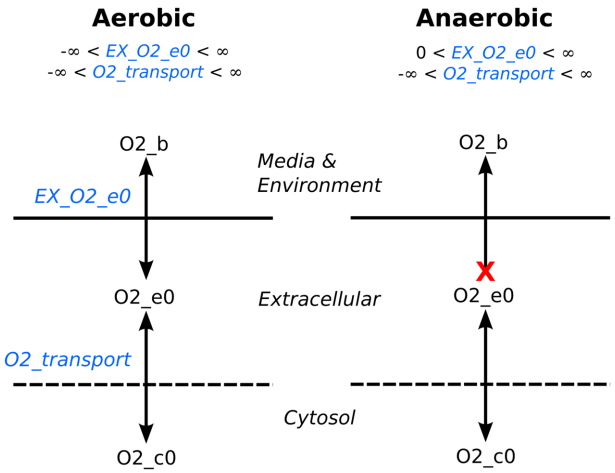


FIGURE 3 | Controlling oxygen exchange example. The lower bound in exchange reactions (e.g., EX_O2_e0) moderate the availability of a compound for the extracellular compartment. To simulate an anaerobic environment, flux from the environment to extracellular space is cutoff by setting the lower bound to zero, whereas the lower bound is set to negative infinity for an aerobic environment, allowing unlimited amount of oxygen into the extracellular space.

they are removed from the media at the same rate that they are created. The reaction bounds can be manipulated to mimic gene knockouts and environmental changes. For example, simulating an anaerobic environment is accomplished by removing the model's input oxygen capability by setting the bound to zero (**Figure 3**). The benefit in this method instead of completely removing the uptake/secretion reaction from the stoichiometric matrix is oxygen produced by intracellular reactions could still be secreted from the cell while still preventing oxygen uptake from the environment.

Compound Bounds

A central tenet of FBA is the assumption that the change in each compound concentration is equal or balanced (**Figure 1C**), and thus overall (with the exception of compounds that are taken up or secreted) the concentration of the compounds sums to 0. This is the steady state mass balance that allows FBA to perform without any kinetic information, and also acts as another constraint on the system. To ensure that the compound consumption is equal to production, the compound bounds are set so that the concentration of the compound can not be greater or less than zero.

The Objective Function (What We Are Trying to Maximize)

A single reaction is designated as the objective function and is included in the stoichiometric matrix. This is usually the biomass reaction (**Supplementary Figure S1**), a complex reaction that consists of biomass precursors (e.g., amino acids, nucleotides, carbohydrates, and lipids; but is almost always dominated by the consumption of ATP) and in return produces *biomass*, a generic term to mean growth of the cell. However, the objective function can consume and produce other cellular components. Likewise, typically the solution required maximizes the flux through the objective function (i.e., produce as much biomass as possible), however, with different outcomes represented by alternate equations it may be more suitable to minimize the flux through that equation. The biomass equation reaction is typically added as the first or last row in the stoichiometric matrix.

Solving the Linear Programming Puzzle

Once the stoichiometric matrix has been constructed, the media defined, the compound and reaction bounds set, and the biomass equation (or other objective function) added to the stoichiometric matrix, the system can be solved by linear programming. The solvers return the value of the flux through the reaction designated as the objective function, with a value above one generally indicating growth, and a value approaching zero indicating no growth (because of the limitations of floating point arithmetic, no growth is not often exactly zero).

Gap-filling

Typically, a genome-scale metabolic model constructed from the annotations in a genome will not result in growth because the metabolic network is incomplete. Additional reactions have to be added to the model to ensure growth, a process called *gap-filling*. The gap-filling step of building the model is the point where most of the erroneous assertions about the metabolism of an organism are made. A selection of reactions has to be added to the metabolic network derived from the genome annotations, and the selection is often made with little supporting evidence. Consequently, there are many different approaches for identifying the reactions missing from a model, including identifying missing reactions from expression information (Kharchenko et al., 2004), finding reactions that complete the metabolic networks based on their topology (Satish Kumar et al., 2007), using phenotypic data to identify growth conditions (Herrgård et al., 2006; Satish Kumar and Maranas, 2009; Vitkin and Shlomi, 2012; Cuevas et al., 2014), and most recently likelihood based approaches that use sequence similarity (Benedict et al., 2014). Enough reactions could be added that result in the growth of any model on any media, but to retain biological accuracy, only those reactions that result in growth on media where the organism is known to grow should be added to the model.

The gap-filling approach taken by most applications is an iterative approach. Starting with the metabolic model derived from the annotations of the genome, the model is tested for growth. While the model does not grow, additional reactions are proposed, based on a variety of criteria, and the model including the additional reactions are tested again. Once the model grows, gap-generation (see below) can be used to prune unnecessary reactions in the model.

PyFBA includes several different modules for gap-filling, sequentially suggesting reactions to add to the model. For example, there is a set of 109 predefined reactions that are present in every model tested to date and that can be added to gap-fill any model. If an organism is known to grow on a particular media, reactions are also identified to be added to

the model that transport the media components into the cell. As noted elsewhere, the identification of transport reactions from genome sequences is problematic, and therefore using phenotype data ensures accurate representation of the biology of the organism in the model. Reactions are identified that connect to orphan compounds – compounds that are only associated with a single reaction in the network. If the orphan compounds are consumed they either need to be produced by another reaction or transported into the cell by a transport reaction. If the orphan compounds are produced in the model they either need to be secreted as waste (via a transport reaction) or consumed in another reaction. A gap-filling approach is also included in PyFBA that analyzes the presence of all the subsystems in the reaction network, and proposes reactions that complete the subsystems in the model. A general framework is also available that accepts tuples of annotations from other genome(s) and the probability that those annotations are associated with the current model and proposes reactions based on those annotations. For example, when building a model of *Citrobacter* a gap-filling approach may be to identify all functional roles annotated in all other *Citrobacter* genomes and the likelihood that the functional role should also be in this *Citrobacter* genome (we typically define this as the fraction of genomes tested that contain this role). Reactions can be suggested for addition to the model based on those annotations and their likelihoods. In addition to suggesting new reactions to the model, the PyFBA gap-filling approach is extensible with new approaches as they are developed.

There are many different approaches that can be used to suggest reactions to be added to models, but most approaches will suggest many more reactions to add than are actually needed for growth. Therefore, excess reactions, that are not absolutely essential for growth, should be trimmed from those suggestions. PyFBA includes a recursive algorithm to reduce a set of proposed reactions from the initial proposal to just those reactions that are absolutely required for growth. Under the best conditions, this algorithm provides $O(\log n)$ complexity (where n is the number of reactions that are proposed to be added) as it is based on binary search, however, under the worst conditions the algorithm will approach $O(n)$ complexity as all the reactions in the model have to be tested.

Gap-generation

Once a model has been generated that grows under a set of conditions, reactions can be recursively removed until the minimum set of reactions that are required for the model to grow are identified. This approach has $O(n)$ complexity (where n is the number of reactions in the model) as each reaction is tested one at a time. It is also worth noting that another method previously developed to identify network redundancy and insubstantial reactions, but is not yet implemented in PyFBA, is flux-variability analysis (Burgard et al., 2001; Gudmundsson and Thiele, 2010). Flux-variability analysis is a linear programming problem that minimizes and maximizes the flux through each reaction while maintaining the same phenotype. Typically, the maximal rate of biomass production is kept constant during flux variable analysis in order to test the flexibility of the metabolic network, or to find ineffective reactions.

Exchanging Models

Several exchangeable file formats exist that can be used to store metabolic models. The most common among existing tools is the systems biology markup language, or SBML (Lakshmanan et al., 2012). JavaScript Object Notation (JSON) is a widespread key-value format used in many web-based software applications. Consequently, JSON libraries are easily accessible in most programming languages. Both formats are supplied as **Supplementary Materials 2** and **3** and are supported by the FBA scripts used here.

CONSTRUCTING GENOME SCALE METABOLIC MODELS USING PyFBA

We have described the principles of how genome-scale metabolic models are created from genome annotations, and in this section we demonstrate how a genome scale metabolic model can be created from genome annotations using PyFBA.

Starting with a genome annotated in RAST, download the annotations as a spreadsheet from the RAST job overview page. This provides a file that can be opened in any office suite of software and contains information about all of the genes identified in the genome, including the location of the protein encoding gene (contig, start, stop, and strand), and most importantly, the function of the protein. We connect the functional roles in the *function* column of this table to the reaction IDs in the Model SEED by using the PyFBA.filters.roles_to_reactions function. This function maps from functional role to reaction ID. We then read an appropriate media file using PyFBA.parse.read_media_file, and either define a new biomass reaction or import one of the predefined reactions from PyFBA.metabolism.biomass.biomass_equation. The stoichiometric matrix is created from these data inputs via the PyFBA.fba.create_stoichiometric_matrix method. The reaction bounds are computed from all the reactions, including the uptake and secretion reactions and the compound bounds (all zero) are added using the two methods PyFBA.fba.reaction_bounds and PyFBA.fba.compound_bounds. Finally, the stoichiometric matrix is solved using the linear programming solver.

Gap-filling is available in PyFBA using the PyFBA.gapfill modules. As described above, these modules provide several different mechanisms for gap-filling as well as methods to bisect the reactions proposed to be added to your model for growth. **Figure 4** illustrates the typical workflow for PyFBA alongside the key module functions used at each step.

We have created an iPython Notebook[7] (**Supplementary Material 4**) that demonstrates each of the steps in creating a model and gap-filling the model using the example data from the *Citrobacter* model discussed previously (Cuevas et al., 2014). A second iPython Notebook[8] (**Supplementary Material 5**) demonstrates how PyFBA can be used to import metabolic models from an SBML document to run FBA.

[7] https://github.com/linsalrob/PyFBA/blob/working/iPythonNotebooks/From_functional_roles_to_gap-filling.ipynb

[8] https://github.com/linsalrob/PyFBA/blob/working/iPythonNotebooks/Using_an_SBML_model.ipynb

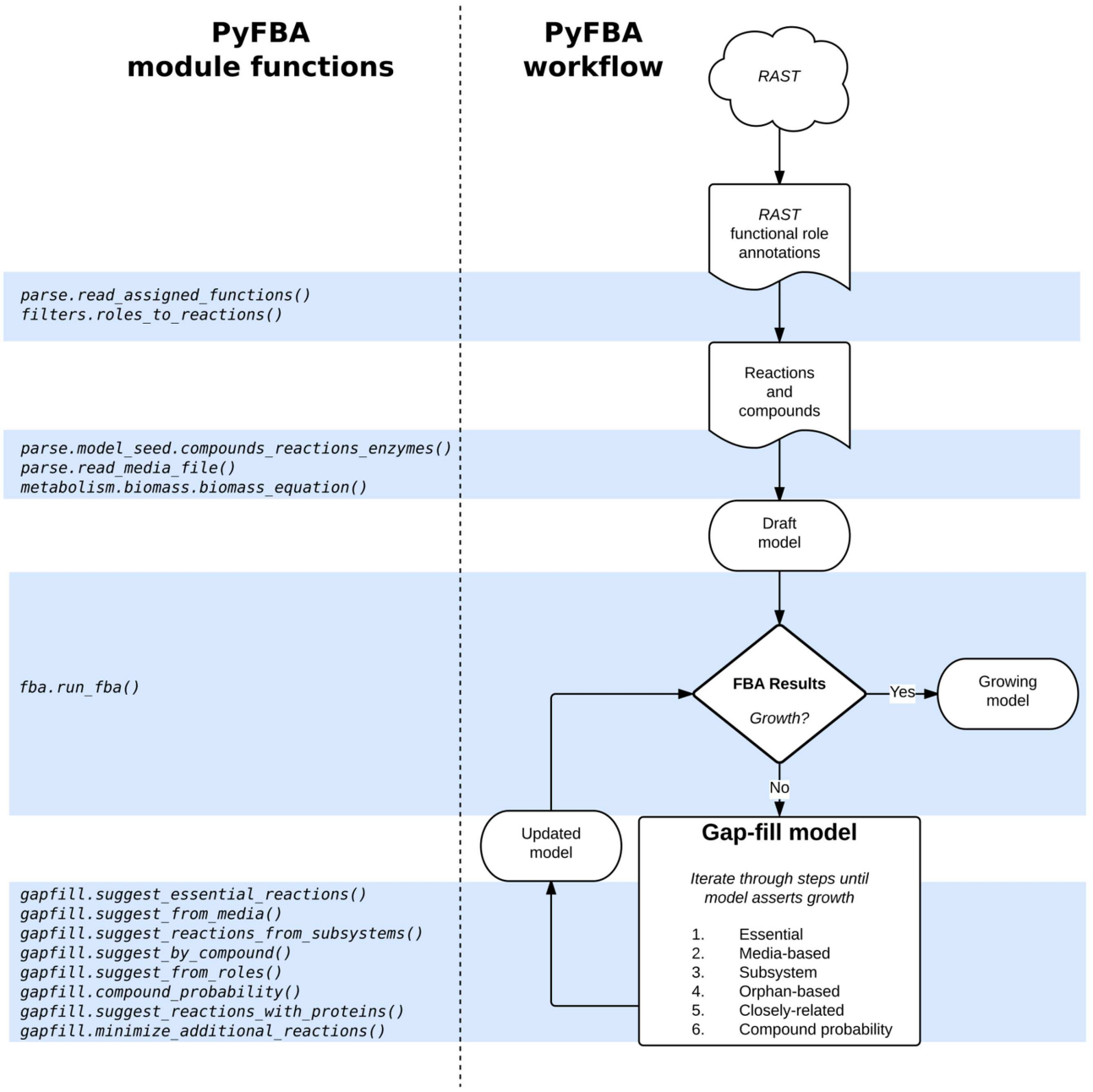


FIGURE 4 | PyFBA workflow. Each step to obtain a genome-scale metabolic model using PyFBA is presented on the right. Arrows represent actions to be taken to obtain the next set of information in the workflow. Corresponding to each step are the necessary PyFBA module functions on the left. The gap-fill step is associated with multiple functions, each one differing by the gap-fill strategy used to identify reactions. The gap-fill method is iterative, hence a subset or all of these modules are used to obtain a growing model. The RAST portion of the workflow is an example for using functional role annotations; these steps can be replaced by other methods yielding the same information.

DISCUSSION

Genome-scale metabolic modeling extends the usefulness of the base microbial genome annotation by identifying the biochemistry that is actually happening in the cell. The conversion of functional roles to reactions highlights our understanding of the cellular metabolism, while the gaps that remain and need filling before the model "grows" highlight areas where we have less detailed understanding of the biochemistry. PyFBA is a novel Python based implementation of flux balance analysis that can be easily extended and modified to provide new metabolic modeling capabilities.

Gap-filling chooses reactions to add to the model in order to generate growth, and with a database of over 30,000 reactions

from which to choose there are many solutions that could result in growth of the model. Therefore, in order to be meaningful, FBA approaches must be constrained using real biological and physicochemical constraints determined from experimentation. Whole genome sequencing and bioinformatics analyses rapidly identify metabolic genes for building draft genome-scale metabolic models (Edwards and Palsson, 2000; Price et al., 2003; Cuevas et al., 2014). Genome-scale databases are expanding and improving, providing more accurate constraints to metabolic models (Overbeek et al., 2013; Wattam et al., 2014; Brettin et al., 2015). More enzymes are being characterized, thus linking new genes to metabolic functions (Kanehisa et al., 2012; Caspi et al., 2014; Sanchez et al., 2015). Accordingly, this will result in more accurate draft metabolic models but will also provide more accurate choices when gap-filling. Transporter reactions are more difficult to differentiate from each other, and some spontaneous reactions do not require any proteins for catalysis. Phenotypic data such as those from minimal media growth experiments provide further evidence to incorporate reactions from particular transporters and enzymes into the metabolic model. Overtime, rendering metabolic models using annotations can be directly improved by the accumulation of unique genomic and phenomic data.

Identifying reactions during gap-filling inadvertently leads to an opportunity to improve genomic annotations. Regions where DNA sequences were not assigned a function due to types of problems with the annotation step, such as the sequence lacking known homology to another organism, can be connected to the gap-filled reactions from PyFBA. This positive feedback has not been efficiently captured by previous software but could provide current databases a means for improving their annotations, narrowing the search space of functional roles the genetic material may be associated with. Future endeavors in model reconciliation will support associations between gap-filled reactions and unannotated genome content.

There are several limitations to the previous versions of genome-scale metabolic modeling that the new iterations of these models overcome. For example, most models lack an integration of regulatory processes in the networks — if a gene is present it is assumed to be expressed and functional. Recent models include transcription factors and gene expression information alongside traditional models (Chandrasekaran and Price, 2010; O'Brien et al., 2013). Another limitation with existing models is the choice of the objective function and the resulting flux distribution through the network. However, the assumption that the flux of compounds through the metabolic networks to optimize growth rate has been shown to correlate with experimental observations (Varma and Palsson, 1993), and maximizing the biomass reaction is consistent with experimental flux data and biomass precursors are in agreement with the experimental observations (Burgard and Maranas, 2003).

AUTHOR CONTRIBUTIONS

DC worked on PyFBA development, algorithm design, prepared and wrote manuscript, created figures, and data analysis. JE provided SEED biochemistry resources and discussion. CH provided SEED resources and discussion. RO provided RAST consultation. TO provided algorithm design discussion. RE worked on PyFBA development, algorithm design, wrote manuscript, data analysis, code testing, and code distribution.

FUNDING

This work is supported by NSF grants CNS-1305112 and MCB-1330800 to ER, and by a STEM scholarship award funded by NSF grant DUE-1259951 to Cuevas.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fmicb.2016.00907

FIGURE S1 | Gram-negative biomass reaction. Coefficients for the reaction are listed before the compound name.

SUPPLEMENTARY MATERIAL 1 | Sample stoichiometric matrix (*Citrobacter sedlakii*). *SM1_stoichiometric_matrix.tsv.*

SUPPLEMENTARY MATERIAL 2 | *Citrobacter sedlakii* SBML file. *SM2_Citrobacter_sedlakii.sbml.*

SUPPLEMENTARY MATERIAL 3 | *Citrobacter sedlakii* JSON file. *SM3_Citrobacter_sedlakii.json.*

SUPPLEMENTARY MATERIAL 4 | iPython Notebook demonstration of PyFBA workflow for converting functional roles to reactions to gap-filling. *SM4_From_functional_roles_to_gap-filling.pdf.*

SUPPLEMENTARY MATERIAL 5 | iPython Notebook demonstration of PyFBA workflow for importing an SBML file and running FBA. *SM5_Using_an_SBML-model.pdf.*

REFERENCES

Aziz, R. K., Bartels, D., Best, A. A., DeJongh, M., Disz, T., Edwards, R. A., et al. (2008). The RAST server: rapid annotations using subsystems technology. *BMC Genomics* 9:75. doi: 10.1186/1471-2164-9-75

Benedict, M. N., Mundy, M. B., Henry, C. S., Chia, N., and Price, N. D. (2014). Likelihood-Based Gene Annotations for Gap Filling and Quality Assessment in Genome-Scale Metabolic Models. *PLoS Comput. Biol.* 10:e1003882. doi: 10.1371/journal.pcbi.1003882

Brettin, T., Davis, J. J., Disz, T., Edwards, R. A., Gerdes, S., Olsen, G. J., et al. (2015). RASTtk: a modular and extensible implementation of the RAST algorithm for building custom annotation pipelines and annotating batches of genomes. *Sci. Rep.* 5:8365. doi: 10.1038/srep08365

Burgard, A. P., and Maranas, C. D. (2003). Optimization-based framework for inferring and testing hypothesized metabolic objective functions. *Biotechnol. Bioeng.* 82, 670–677. doi: 10.1002/bit.10617

Burgard, A. P., Vaidyaraman, S., and Maranas, C. D. (2001). Minimal reaction sets for *Escherichia coli* metabolism under different growth requirements and uptake environments. *Biotechnol. Prog.* 17, 791–797. doi: 10.1021/bp0100880

Carrera, J., Estrela, R., Luo, J., Rai, N., Tsoukalas, A., and Tagkopoulos, I. (2014). An integrative, multi-scale, genome-wide model reveals the phenotypic landscape of *Escherichia coli*. *Mol. Syst. Biol.* 10, 735. doi: 10.15252/msb.20145108

Caspi, R., Altman, T., Billington, R., Dreher, K., Foerster, H., Fulcher, C. A., et al. (2014). The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic Acids Res.* 42, D459–D471. doi: 10.1093/nar/gkt1103

Chandrasekaran, S., and Price, N. D. (2010). Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in *Escherichia Coli* and *Mycobacterium Tuberculosis*. *Proc. Natl. Acad. Sci.* 107, 17845–17850. doi: 10.1073/pnas.1005139107

Conesa, A., Götz, S., García-Gómez, J. M., Terol, J., Talón, M., and Robles, M. (2005). Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics* 21, 3674–3676. doi: 10.1093/bioinformatics/bti610

Cuevas, D. A., Garza, D., Sanchez, S. E., Rostron, J., Henry, C. S., Vonstein, V., et al. (2014). Elucidating genomic gaps using phenotypic profiles. *F1000Research* 3:210. doi: 10.12688/f1000research.5140.1

Cuevas, D. A., O'Connell, T. G., and Edwards, R. A. (2016a). "PyFBA." *GitHub repository*. https://github.com/linsalrob/PyFBA.

Cuevas, D. A., O'Connell, T. G., and Edwards, R. A. (2016b). "PyFBA." *Python Package Index repository*. https://pypi.python.org/pypi/PyFBA.

DeJongh, M., Formsma, K., Boillot, P., Gould, J., Rycenga, M., and Best, A. (2007). Toward the automated generation of genome-scale metabolic networks in the seed. *BMC Bioinformatics* 8:139. doi: 10.1186/1471-2105-8-139

Devoid, S., Overbeek, R., DeJongh, M., Vonstein, V., Best, A. A., and Henry, C. (2013). "Automated genome annotation and metabolic model reconstruction in the SEED and model SEED," in *Systems Metabolic Engineering*, ed. H. S. Alper (Totowa, NJ: Humana Press), 17–45. doi: 10.1007/978-1-62703-299-5_2

Ebrahim, A., Lerman, J. A., Palsson, B. O., and Hyduke, D. R. (2013). COBRApy: constraints-based reconstruction and analysis for python. *BMC Syst. Biol.* 7: 74. doi: 10.1186/1752-0509-7-74

Edwards, J. S., Covert, M., and Palsson, B. (2002). Metabolic modelling of microbes: the flux-balance approach. *Environ. Microbiol.* 4, 133–140. doi: 10.1046/j.1462-2920.2002.00282.x

Edwards, J. S., and Palsson, B. Ø. (2000). The *Escherichia Coli* MG1655 in silico metabolic genotype: its definition, characteristics, and capabilities. *Proc. Natl. Acad. Sci.* 97, 5528–5533. doi: 10.1073/pnas.97.10.5528

Gasteiger, E., Gattiker, A., Hoogland, C., Ivanyi, I., Appel, R. D., and Bairoch, A. (2003). ExPASy: The proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res.* 31, 3784–3788. doi: 10.1093/nar/gkg563

Gudmundsson, S., and Thiele, I. (2010). Computationally efficient flux variability analysis. *BMC Bioinformatics* 11:489. doi: 10.1186/1471-2105-11-489

Hamilton, J. J., and Reed, J. L. (2014). Software platforms to facilitate reconstructing genome-scale metabolic networks. *Environ. Microbiol.* 16, 49–59. doi: 10.1111/1462-2920.12312

Herrgård, M. J., Fong, S. S., and Palsson, B. Ø. (2006). Identification of genome-scale metabolic network models using experimentally measured flux profiles. *PLoS Comput. Biol.* 2:e72. doi: 10.1371/journal.pcbi.0020072

Hoppe, A., Hoffmann, S., Gerasch, A., Gille, C., and Holzhütter, H. G. (2011). "FASIMU: flexible software for flux-balance computation series in large metabolic networks. *BMC Bioinformatics* 12:28. doi: 10.1186/1471-2105-12-28.

IBM ILOG (2014). *Cplex Optimization Studio*. Available at: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

Kanehisa, M., Goto, S., Kawashima, S., Okuno, Y., and Hattori, M. (2004). The KEGG resource for deciphering the genome. *Nucleic Acids Res.* 32(suppl 1): D277–D280. doi: 10.1093/nar/gkh063

Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., and Tanabe, M. (2012). KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Res.* 40: D109–D114. doi: 10.1093/nar/gkr988.

Kauffman, K. J., Prakash, P., and Edwards, J. S. (2003). Advances in flux balance analysis. *Curr. Opin. Biotechnol.* 14, 491–96. doi: 10.1016/j.copbio.2003.08.001

Kharchenko, P., Vitkup, D., and Church, G. M. (2004). Filling gaps in a metabolic network using expression information. *Bioinformatics* 20(suppl 1): i178–i185. doi: 10.1093/bioinformatics/bth930

Klamt, S., and von Kamp, A. (2011). "An application programming interface for CellNetAnalyzer. Biosystems," in *Proceedings of the workshop Integration of OMICS Datasets into Metabolic Pathway Analysis,* Vol.105, Edinburgh, 162–168. doi: 10.1016/j.biosystems.2011.02.002

Lakshmanan, M., Koh, G., Chung, B. K., and Lee, D. Y. (2012). Software applications for flux balance analysis. *Brief. Bioinform.* 108–122. doi: 10.1093/bib/bbs069

Lee, J. M., Gianchandani, E. P., and Papin, J. A. (2006). Flux balance analysis in the era of metabolomics. *Brief. Bioinform.* 7, 140–150. doi: 10.1093/bib/bbl007

Liu, J. K., Brien, E. J. O. Lerman, J. A., Zengler, K., Palsson, B. O., and Feist, A. M. (2014). Reconstruction and modeling protein translocation and compartmentalization in *Escherichia Coli* at the genome-scale. *BMC Sys. Biol.* 8: 110. doi: 10.1186/s12918-014-0110-6

Makhorin, A. (2008). "GLPK (GNU linear programming kit),".

Marger, M. D., and Saier, M. H. (1993). A major superfamily of transmembrane facilitators that catalyse uniport, symport and antiport. *Trends in Biochem. Sci.* 18:13–20. doi: 10.1016/0968-0004(93)90081-W

MATLAB (2012). *MATLAB 8.0 and Statistics Toolbox 8.1*. Natick, MA: The MathWorks, Inc.

O'Brien, E. J., Lerman, J. A., Chang, R. L., Hyduke, D. R., and Palsson, B. Ø. (2013). Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Mol. Sys. Biol.* 9, 693. doi: 10.1038/msb.2013.52

Oberhardt, M. A., Puchałka, J., Martins dos Santos, V. A., and Papin, J. A. (2011). Reconciliation of genome-scale metabolic reconstructions for comparative systems analysis. *PLoS Comput Biol* 7:e1001116. doi: 10.1371/journal.pcbi.1001116

Orth, J. D., Thiele, I., and Palsson, B. Ø. (2010). What is flux balance analysis? *Nature Biotechnol.* 28, 245–48. doi: 10.1038/nbt.1614

Overbeek, R., Larsen, N., Pusch, G. D., D'Souza, M., Selkov, E Jr., Kyrpides, N., et al. (2000). WIT: Integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Res.* 28, 123–25. doi: 10.1093/nar/28.1.123

Overbeek, R., Larsen, N., Walunas, T., D'Souza, M., Pusch, G., Selkov, E Jr., et al. (2003). The ergotm genome analysis and discovery system. *Nucleic Acids Res.* 31, 164–71. doi: 10.1093/nar/gkg148

Overbeek, R., Olson, R., Pusch, G. D., Olsen, G. J., Davis, J. J., Disz, T., et al. (2013). The SEED and the rapid annotation of microbial genomes using subsystems technology (RAST). *Nucleic Acids Res.* 42, D206–D214. doi: 10.1093/nar/gkt1226

Palsson, B. (2000). The challenges of in silico biology. *Nature Biotechnol.* 18, 1147–1150. doi: 10.1038/81125

Plata, G., Henry, C. S., and Vitkup, D. (2015). Long-term phenotypic evolution of bacteria. *Nature* 517, 369–372. doi: 10.1038/nature13827

Price, N. D., Papin, J. A., Schilling, C. H., and Palsson, B. O. (2003). Genome-scale microbial in silico models: the constraints-based approach. *Trends Biotechnol.* 21, 162–169. doi: 10.1016/S0167-7799(03)00030-1

Raman, K., and Chandra, N. (2009). Flux balance analysis of biological systems: applications and challenges. *Brief. Bioinform.* 10, 435–449. doi: 10.1093/bib/bbp011

Saier, M. H. (1994). Computer-aided analyses of transport protein sequences: gleaning evidence concerning function, structure, biogenesis, and evolution. *Microbiol. Rev.* 58, 71–93.

Sanchez, S. E., Cuevas, D. A., Rostron, J. E., Liang, T. Y., Pivaroff, C. G., Haynes, M. R., et al. (2015). Phage phenomics: physiological approaches to characterize novel viral proteins. *J. Vis. Exp.* 100, e52854. doi: 10.3791/52854

Satish Kumar, V., Dasika, M. S., and. Maranas, C. D. (2007). Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics* 8:212. doi: 10.1186/1471-2105-8-212

Satish Kumar, V., and Maranas, C. D. (2009). GrowMatch: an automated method for reconciling *in silico/in vivo* growth predictions. *PLoS Comput. Biol.* 5:e1000308. doi: 10.1371/journal.pcbi.1000308

Schellenberger, J., Que, R., Fleming, R. M., Thiele, I., Orth, J. D., Feist, A. M., Zielinski, D. C., et al. (2011). Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox v2.0. *Nat. Protoc.* 6, 1290–1307. doi: 10.1038/nprot.2011.308

Schomburg, I., Chang, A., and Schomburg, D. (2002). BRENDA, Enzyme data and metabolic information. *Nucleic Acids Res.* 30, 47–49. doi: 10.1093/nar/30.1.47

Seaver, S. M., Henry, C. S., and Hanson, A. D. (2012). Frontiers in metabolic reconstruction and modeling of plant genomes. *J. Exp. Bot.* 63, 2247–2258. doi: 10.1093/jxb/err371

Seemann, T. (2014). Prokka: rapid prokaryotic genome annotation. *Bioinformatics* 30, 2068–2069. doi: 10.1093/bioinformatics/btu153

Tobes, R., Pareja-Tobes, P., Manrique, M., Pareja-Tobes, E., Kovach, E., Alekhin, A., et al. (2015). "Gene calling and bacterial genome annotation with BG7," in *Bacterial Pangenomics*, eds A., Mengoni, M., Galardini, and M., Fondi (New York, NY: Springer), 177–189. doi: 10.1007/978-1-4939-1720-4_12

Van Domselaar, G. H., Stothard, P., Shrivastava, S., Cruz, J. A., Guo, A., and Dong, X., (2005). BASys: a web server for automated bacterial genome annotation. *Nucleic Acids Res.* 33(suppl. 2), W455–W459. doi: 10.1093/nar/gki593

Varma, A., and Palsson, B. Ø. (1993). Metabolic capabilities of *Escherichia Coli*. II: optimal growth patterns. *J. Theor. Biol.* 165, 503–522. doi: 10.1006/jtbi.1993.1203

Vitkin, E., and Shlomi, T. (2012). MIRAGE: a functional genomics-based approach for metabolic network model reconstruction and its application to cyanobacteria networks. *Genome Biol.* 13:R111. doi: 10.1186/gb-2012-13-11-r111

Wattam, A. R., Abraham, D., Dalay, O., Disz, T. L., Driscoll, T., Gabbard, J. L., et al. (2014). PATRIC, the bacterial bioinformatics database and analysis resource. *Nucleic Acids Res.* 42, D581–D591. doi: 10.1093/nar/gkt1099

Webb, E. C. (1992). Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes. Academic Press Available at: http://www.cabdirect.org/abstracts/19930457289.html.

Wright, J., and Wagner. A. (2008). The systems biology research tool: evolvable open-source software. *BMC Syst. Biol.* 2:55. doi: 10.1186/1752-0509-2-55

Yurkovich, J. T., and Palsson, B. Ø. (2016). Solving Puzzles With Missing Pieces: The Power of Systems Biology. *Proc. IEEE* 104, 2–7. doi: 10.1109/JPROC.2015.2505338

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.