Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox

Scott A Becker, Adam M Feist, Monica L Mo, Gregory Hannum, Bernhard Ø Palsson & Markus J Herrgard

Department of Bioengineering, University of California San Diego, 9500 Gilman Dr., La Jolla, California 92093-0412, USA. Correspondence should be adressed to M.J.H. (mherrgar@ucsd.edu).

Published online 29 March 2007; doi:10.1038/nprot.2007.99

The manner in which microorganisms utilize their metabolic processes can be predicted using constraint-based analysis of genome-scale metabolic networks. Herein, we present the constraint-based reconstruction and analysis toolbox, a software package running in the Matlab environment, which allows for quantitative prediction of cellular behavior using a constraint-based approach. Specifically, this software allows predictive computations of both steady-state and dynamic optimal growth behavior, the effects of gene deletions, comprehensive robustness analyses, sampling the range of possible cellular metabolic states and the determination of network modules. Functions enabling these calculations are included in the toolbox, allowing a user to input a genome-scale metabolic model distributed in Systems Biology Markup Language format and perform these calculations with just a few lines of code. The results are predictions of cellular behavior that have been verified as accurate in a growing body of research. After software installation, calculation time is minimal, allowing the user to focus on the interpretation of the computational results.

INTRODUCTION

Systems biology is a rapidly growing field[1] that is based on building and validating *in silico* models of biological systems using a wealth of experimental data. These models can be applied to generate novel, testable and often quantitative predictions of cellular behavior. Genome-scale network reconstruction efforts have been employed to generate models of diverse cellular processes such as signal transduction[2], transcriptional regulation[3,4] and metabolism[5]. A reconstruction is herein defined as the list of biochemical reactions occurring in a particular cellular system (such as metabolism) and the associations between these reactions and relevant proteins, transcripts and genes. A reconstruction can be converted to a model by including the assumptions necessary for computational simulation, for example, maximum reaction rates and nutrient uptake rates. An extensive array of methods for analyzing these kinds of genome-scale models have been developed and the methods have been applied to study a growing number of biological questions[6–10]. The constraint-based reconstruction and analysis (COBRA) strategy is depicted in **Figure 1**.

As reconstructed networks have been made publicly available by our group (http://systemsbiology.ucsd.edu/organisms) and other groups[11–14], researchers around the world have undertaken novel computational studies utilizing the networks (see e.g., http://systemsbiology.ucsd.edu/organisms/ecoli/ecoli_others.html). Many studies apply a core set of basic *in silico* methods and often also describe novel methods to interrogate different models. Despite the large number of studies utilizing reconstructed models, there is no single freely available software toolbox that would allow easy application of all these methods. This protocol aims to make *in silico* analysis of these networks accessible to researchers with a wide range of interests, particularly those who do not have expertise in the specific types of computations presented here. We have produced an integrated set of software tools that allows application of *in silico* methods that are commonly used to analyze reconstructed metabolic networks. Although we will only discuss the application of *in silico* methods to genome-scale metabolic networks, the majority of the methods described here can also be applied to study network properties of signaling network reconstructions[2]. Furthermore, in principle, constraint-based methods are applicable to any system that can be represented as a set of chemical reactions.

The methods we describe inherently have limitations, as is true of all computational predictions, and the results of a simulation are best understood as hypotheses. For example, given the enormous quantity of double gene deletions possible in yeast, a model can in a short time frame predict which would reduce growth by between 50% and 90%, but the hypotheses about epistatic interactions between genes must be experimentally verified. However, considering the low frequency of epistatic interactions discovered experimentally[15], the model-based approach provides a valuable and reasonably sized starting set of hypothetical interactions for experimental testing. Most of the methods described herein compute reaction fluxes, which are a quantitative representation of the reaction rates of each biochemical reaction in the network. In general, especially with larger metabolic networks, the fluxes within a cell cannot be uniquely calculated because a range of feasible values exist when fluxes are subjected to known constraints.

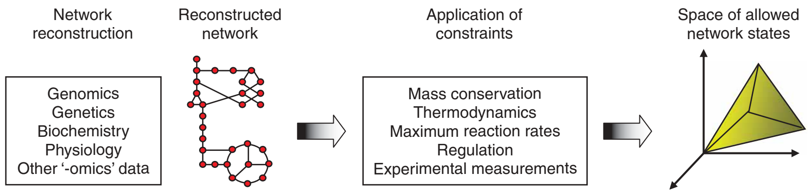
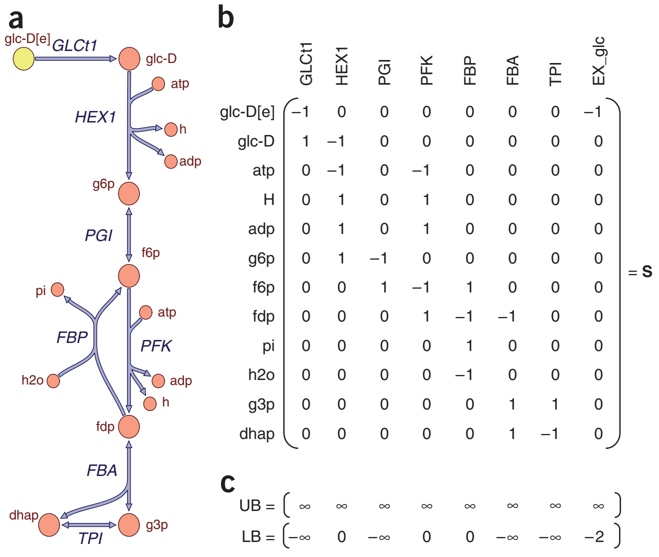


Figure 1 | COBRA. A number of different types of physical, chemical and biological constraints can be used to define the range of potential states a biochemical reaction network can achieve. In the case of metabolism, these states correspond to different flux distributions through the metabolic network.



$$\begin{array}{c|cccccccc}
 & \text{GLCt1} & \text{HEX1} & \text{PGI} & \text{PFK} & \text{FBP} & \text{FBA} & \text{TPI} & \text{EX\_glc} \\
\hline
\text{glc-D[e]} & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\text{glc-D} & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{atp} & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
\text{H} & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{adp} & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{g6p} & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
\text{f6p} & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 \\
\text{fdp} & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 \\
\text{pi} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\text{h2o} & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
\text{g3p} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\text{dhap} & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0
\end{array} = \mathbf{S}$$

$$\text{UB} = \begin{pmatrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

$$\text{LB} = \begin{pmatrix} -\infty & 0 & -\infty & 0 & 0 & -\infty & -\infty & -2 \end{pmatrix}$$

Figure 2 | Stoichiometric representation of metabolic networks. (**a**) The first few reactions of glycolysis in a graphical form. (**b**) The stoichiometric matrix (**S**) corresponding to **a**. As indicated, each column corresponds to a particular reaction and each row to a particular metabolite. The last column, labeled "EX_glc," is an exchange reaction for glucose that allows glucose to enter and leave the system. (**c**) The upper (UB) and lower bounds (LB) for each reaction. The three reversible reactions (PGI, FBA and TPI) have lower bounds of $-\infty$. The irreversible reactions have lower bounds of zero because they are not to proceed in the reverse direction. The exchange reaction in the last column has a lower bound of "-2" indicating a potential glucose uptake rate of 2 mmol gDW^{-1} h^{-1}. All reactions are effectively unconstrained in the forward direction.

In addition, some of the methods we describe are based on the assumption that cells strive to maximize their growth rate. This assumption is satisfied by simulating maximal production of the molecules required to make new cells (biomass precursor molecules). The maximum growth rate assumption is not always true, but it provides an acceptable starting point for many types of computations. In spite of their limitations, the predictive power of genome-scale models of metabolic networks has been demonstrated in diverse situations through careful experimentation (see for example refs. 16–19).

Before calculation, pertinent details of the reconstructed metabolic network must be represented mathematically. The stoichiometric matrix, **S**, is the centerpiece of a mathematical representation of genome-scale metabolic networks[5]. This matrix represents each reaction as a column and each metabolite as a row, where each numerical element is the corresponding stoichiometric coefficient. **Figure 2a** shows a graphical form of the first few reactions of glycolysis and **Figure 2b** shows the corresponding stoichiometric matrix. Given a set of reactions, the matrix is constructed in a straightforward and unambiguous manner. It is important to note that if the same compound exists in multiple cellular compartments (for example, ATP is present in both the cytosol and mitochondria in eukaryotes), it must be given a separate row for each compartment. These rows are not interchangeable and the same compound in two different compartments is treated as two distinct compounds (which may be linked by a transport reaction, if physiologically appropriate). In a compartmentalized network, difficulty can be avoided by uniquely naming compounds *compound_abbreviation[compartment_abbreviation]* for all compartments other than the cytosol. The metabolites glc-D and glc-D[e] demonstrate this concept in **Figure 2a,b**, where the transport reaction GLCt1 connects the two metabolites.

In addition to the stoichiometric matrix, all protocols described herein require defining an upper and lower bound for the allowable flux through each reaction. Biologically speaking, this represents the lowest and highest reaction rate possible for each reaction. In many cases, reversible reactions within the cell are defined to have an arbitrary large upper bound and an arbitrarily large negative lower bound. Irreversible reactions have a lower bound that is nonnegative, usually zero. Upper and lower bounds are shown schematically in **Figure 2c**. The set of upper and lower bounds is represented as two separate vectors, each containing as many components as there are columns in the stoichiometric matrix, and in the same order.

In order for the predicted fluxes to be meaningful, at least one of the reactions in the model must have a constrained lower/upper bound. Typically, the substrate (e.g., glucose or oxygen) uptake rates are set to experimentally measured values so that most fluxes in the model are constrained by substrate availability. For this reason, setting upper and lower bounds is especially important for reactions that exchange metabolites across the system boundary (i.e., exchange reactions). Exchange reactions serve to uptake compounds (e.g., glucose; see the last column of **Fig. 2b**) to the cell or secrete compounds (e.g., lactic acid) from the cell. If glucose is to be provided to the cell in restricted quantities, the lower bound of its exchange reaction column must be a finite negative number using this orientation (last column of **Fig. 2c**). If a compound is allowed to leave the system, its upper bound must be greater than zero. Taken together, the upper and lower bounds for exchange reactions are quantitative *in silico* representations of the (often experimentally determined) growth media conditions.

All the protocols described below are based on an integrated toolbox (named COBRA Toolbox; http://systemsbiology.ucsd.edu/downloads/COBRAToolbox/) of functions that is used within the Matlab (The Mathworks Inc.) numerical computation and visualization environment. Matlab allows straightforward implementation of new methods and easy modification of existing methods with its scripting capabilities. The advanced analysis and visualization capabilities of Matlab allow users to manipulate and scrutinize the large-scale data sets created by most of the *in silico* methods described below. Furthermore, the COBRA Toolbox can easily be interfaced with other software tools for analyzing metabolic networks and fluxes in Matlab that complement the COBRA Toolbox[20,21]. Although the COBRA Toolbox could (with some modification) handle any reasonable input format for the models, we describe the model input using the Systems Biology Markup Language (SBML) format[22]; we distribute all of our models in this format.

Advanced users will find that the toolbox distribution contains far more potential than is explicitly described in this protocol. Examination of the help output (as detailed under EQUIPMENT SETUP) will reveal optional parameters that may suit some users research interests and needs better.

A graphical overview of the workflow and function names is provided in **Figure 3**.

MATERIALS

EQUIPMENT

- A standard personal computer capable of running Matlab
- Version 6.0 or above of Matlab (Mathworks Inc.) numerical computation and visualization software (http://www.mathworks.com)
- The COBRA Toolbox (the most up-to-date version is provided at http://systemsbiology.ucsd.edu/downloads/COBRAToolbox/)
- The SBML Toolbox for Matlab to allow reading models in SBML format (http://sbml.org/software/sbmltoolbox/)
- A linear programming (LP) solver. Currently, COBRA Toolbox allows using five different solvers:
  - lp_solve: https://sourceforge.net/projects/lpsolve
  - glpk: http://www.gnu.org/software/glpk/
  - LINDO (LINDO Systems Inc.) Matlab API: http://www.lindo.com
  - CPLEX (ILOG Inc.) through the Tomlab (Tomlab Optimization Inc.) optimization environment: http://tomopt.com/
  - Mosek (MOSEK ApS): http://www.mosek.com
- Optional: Cytoscape[23] network visualization software that can be used to draw subnetworks (http://www.cytoscape.org)

EQUIPMENT SETUP

COBRA Toolbox The methods provided in the COBRA Toolbox can be used, in principle, on any metabolic network, but the more computationally intensive calculations may require extensive computer time. The metabolic network models should be represented in SBML format and the model parameters (upper/lower bounds, gene–reaction associations) within the SBML file as indicated in **Supplementary Information**. The COBRA Toolbox consists of files, which should be placed in a local folder on the user's computer. SBML files describing the models should also be stored in a local folder, in order to allow access to the models. Full documentation of the functions described below is available through Matlab's "help" facility by typing "help function_name" on Matlab command line. The same documentation is also included in the form of html files in the "doc" subfolder within the main Toolbox folder, as well as on http://systemsbiology.ucsd.edu/downloads/COBRAToolbox/

SBML file Documentation on the format of the files and how to set up a particular model in this format is provided on the official SBML website (http://sbml.org/documents/, level 2 version 1) and in **Supplementary Information**. The SBML file describing the model must include the following information for all calculations: stoichiometry of each reaction, upper/lower bounds of each reaction and objective function coefficients for each reaction. In addition, the gene deletion studies require gene–reaction associations to be included in the "Notes" section of the SBML file. See the included models (**Supplementary Data 1** and **2**) as guides.

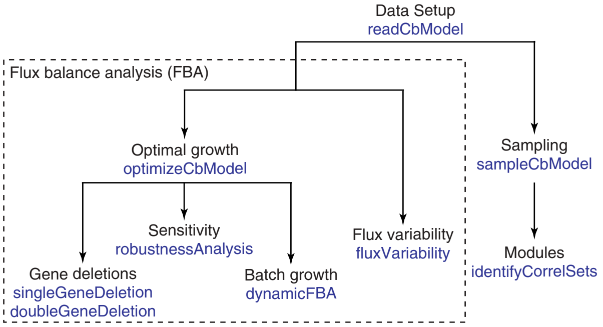


Figure 3 | The workflow for using the COBRA Toolbox. Functions at lower levels of the hierarchy rely on the internal implementation of the methods at higher levels. For example, the gene deletion functions perform an optimal growth calculation for each possible metabolic network with the reaction(s) associated with one or two genes removed. The exception to this is that finding network modules requires the user to input a previously calculated set of samples from the sampling function. On the whole, most of the methods rely on the principles of FBA.

Variables The values that are computed by the toolbox are fluxes, which can be best understood as reaction rates. The units for fluxes used throughout this protocol are mmol gDW^{-1} h^{-1}, where gDW is the dry weight of the cell in grams. The biomass functions included in each SBML file are weighted combinations of molecules that are required for cellular growth and reproduction and are scaled such that the units are h^{-1}. Concentrations are expressed in units of mmol gDW^{-1} ml^{-1}, where the unit of volume is arbitrary.

Installation The Matlab software, SBML Toolbox and one or more of the suggested LP solvers should be installed following the software provider's instructions. Note that the SBML Toolbox and the LP solver need to be accessible in the Matlab path. Sample installation instructions for the lp_solve LP solver on Windows are included in **Supplementary Information**. Users wishing to use Cytoscape for visualization of metabolic networks must install Cytoscape separately. ▲ CRITICAL The SBML Toolbox and the LP solver should be tested for functionality following the software provider's instructions before attempting to use the COBRA Toolbox.

PROCEDURE

Initializing the toolbox

1| Install Matlab, the required toolboxes (SBML Toolbox and COBRA Toolbox) and an LP solver. Start Matlab as described in the installation instructions. Within Matlab, move to the directory where the COBRA Toolbox was installed. Add the directories to the current path by issuing the command initCobraToolbox from the Matlab command line. Note that the default LP solver can be changed by editing the initCobraToolbox script or at any time during a Matlab session by using the changeCobraSolver function included in the Toolbox.

Reading SBML format models into Matlab

2| The following Matlab function from the COBRA Toolbox is used for reading SBML files, assuming the file is in the current working directory:

```
model = readCbModel(fileName);
```

The resulting loaded metabolic model is stored as a structure named model in Matlab. This structure contains the necessary fields to describe the model, including gene–reaction associations (see **Supplementary Information**).

▲ CRITICAL STEP If this step fails, none of the following functions can be used. The problem is either with accessing the SBML files describing the model, the format of the SBML file (if not using files provided with the COBRA Toolbox) or installation/access to the SBML Toolbox. Carefully ensure that the SBML Toolbox is functional and included in the Matlab path, and that the SBML file is correctly formatted (see **Supplementary Information**) and is located in the current Matlab working directory.

Changing lower and/or upper bounds of flux through reactions in the model

3| This is typically used to simulate different media conditions, because setting a negative value for the lower bound of an exchange reaction will set the maximum uptake rate of that metabolite into the system. The relevant function is:

model = changeRxnBounds(model, rxnNameList, boundValue, boundType);

The list of reactions whose bounds are to be changed is described in rxnNameList. The vector boundValue contains the new lower or upper bounds to be assigned to the reactions in reactionList. For exchange reactions, negative lower bounds allow entry into the system, whereas positive upper bounds allow exit from the system. The parameter boundType defines which bound(s) to change to boundValue. There are three possible values for this parameter: "l," lower bound; "u," upper bound; and "b," both bounds set to same value (set the flux to a constant value).

The same function can be used to change the bounds on an internal reaction to match known data or to adjust the reversibility of a reaction. For example, if a reaction rate is experimentally known to always fall within a certain range, its bounds can be set accordingly. Similarly, if a reaction is always observed to proceed in the forward direction in experiments, its lower bound can be set to zero.

Changing the objective function in the model

4| In constraint-based modeling approaches that utilize optimization, a cellular objective function needs to be defined in order to find flux distributions that optimize this objective. The function to change the objective is:

model = changeObjective(model,rxnNameList,objectiveCoeff)

The reactions whose objective coefficients are changed are listed in rxnNameList. The objectiveCoeff vector contains the objective coefficients for each reaction in rxnNameList.

Adding or removing reactions

5| The COBRA Toolbox includes a function to add a new reaction to a model:

model = addReaction(model,rxnName,metaboliteList,stoichCoeffList);

metaboliteList and stoichCoeffList contain the names of the metabolites participating in the reaction (name designated in rxnName) and the corresponding stoichiometric coefficients in a vector. By default, this function adds a reversible reaction that does not participate in the cellular objective function and has no gene associations. The function to remove one or more reactions is:

model = removeRxns(model,rxnRemoveList);

Here, rxnRemoveList contains the names of the reactions to be removed from the model. By default, metabolites that are no longer used by any of the reactions left in the model are also removed. Note that this function does not check whether the model is still functional after removing the reactions the user designates.

Printing out reaction formulas

6| The COBRA Toolbox allows the user to print out the stoichiometric formula for any reaction in the model using the following function:

printRxnFormula(model,rxnNameList);

Here, rxnNameList contains the names of one or more reactions included in the model.

Simulating maximal growth using flux-balance analysis (FBA)

7| One of the most fundamental genome-scale phenotypic calculations is the simulation of cellular growth using FBA. Given an uptake rate for key nutrients (such as glucose and oxygen) and the biomass composition of the cell (usually in mmol component gDW^{-1} and defined in the biomass objective function), the maximum possible growth rate of the cells can be predicted *in silico*. FBA is based on linear optimization of an objective function, which typically is biomass formation. Two functions in the COBRA Toolbox solve this problem and display the results.

solution=optimizeCbModel(model);

This function performs FBA on the model in order to maximize the current objective function. The result is the data structure solution, which contains an optimal solution for the model that was input. The field "*x*" describes a particular, possibly non-unique, optimal flux distribution through the network. The field "*f*" gives the objective value, which corresponds to the predicted unique optimal growth rate.

printFluxVector(model, solution.x,nonZeroFlag,excFlag);

This function takes the variable "*x*" from the structure solution and prints it out to the screen. The output will contain a list of all reactions in the model next to their corresponding flux value. Setting the nonZeroFlag to "true" indicates that only non-zero fluxes should be reported, and setting the excFlag to "true" indicates that only exchange fluxes should be reported. For a specific example, including the expected results, see ANTICIPATED RESULTS. Lower bounds on exchange reactions can be changed to simulate different media conditions. An example provided in the ANTICIPATED RESULTS section illustrates the effect of supplying fructose in addition to glucose in minimal medium.

▲ CRITICAL STEP All the remaining steps of the protocol depend on the functionality of the optimizeCbModel function. If the function does not return a feasible flux distribution for the example cases described below, the problem is most likely with the installation of the LP solver. The functionality of the solver should be tested separately following the software provider's instructions (typically, an example problem is provided with the installation package) before attempting to use it within the COBRA Toolbox. After testing the functionality of the solver within Matlab, it is also necessary to ensure that the relevant solver directories are included in the Matlab path.

Robustness analysis

8| The effect of reducing flux through a single reaction on growth is a network property of great interest. One could, for example, study the effect of decreasing the expression level of a specific metabolic enzyme on the growth rate in order to predict haploinsufficient phenotypes in eukaryotes. Robustness analysis allows the computation of how an objective of interest (e.g., growth rate) changes as the flux through a specific reaction of interest varies in magnitude. The function used for a robustness analysis is:

```
robustnessAnalysis(model, controlRxn, nPoints)
```

This function is used to compute and plot the value of the model objective function as a function of flux values for a reaction of interest (controlRxn) as a means to analyze the network robustness with respect to that reaction. The range of reaction values vary within the minimum and maximum bounds for the reaction of interest and the objective function is maximized using FBA according to each set flux value of controlRxn. A plot with a defined number of points (nPoints) can be generated to visually assess how the objective function changes as the flux through the control reaction varies.

Dynamic FBA growth simulations (batch growth simulations)

9| FBA analyzes network capabilities under a steady-state assumption. Additionally, FBA can be used to examine dynamic processes such as microbial growth in batch cultures by combining FBA with an iterative approach based on a quasi-steady-state assumption (static optimization–based dynamic FBA). At each time step, FBA is used to predict growth, nutrient uptake and by-product secretion rates. These rates are then used to calculate biomass and nutrient concentrations in the culture at the end of the time step. The concentrations can, in turn, be used to calculate maximum uptake rates of nutrients for the next time step. Using this iterative procedure, dynamic FBA has allowed the simulation of both batch and fed-batch experiments[24]. The function is called as:

```
dynamicFBA(model, substrateRxns, initConcentrations, initBiomass, tStep, nSteps,plotRxns)
```

This function will perform dynamic FBA that can be used to predict the outcomes of growth in batch culture conditions. The list of exchange reactions corresponding to the substrates that are initially in the media (e.g., glucose, ammonia, phosphate) is described in substrateRxns. The initConcentrations variable sets the initial concentrations of substrates in the substrateRxns vector. The initBiomass variable is needed to specify the initial amount of biomass in the simulation. The tStep variable sets the time step size interval (h) and the nSteps variable designates the maximum number of time steps for the analysis. The plotRxns variable is optional and contains the names of the exchange reactions for the metabolites whose time-dependent concentrations should be plotted graphically. In some cases there can be equivalent optimal solutions for a given condition (e.g., equivalent excretion profiles). Therefore, flux variability analysis (FVA; see below) should be used to check for the uniqueness of an optimal solution.

Simulating gene deletion phenotypes and epistatic interactions

10| The effect of a gene deletion experiment on cellular growth can be simulated in a manner similar to linear optimization of growth as illustrated above. The upper and lower flux bounds for the reaction(s) corresponding to the deleted gene indicated by its gene–reaction association are both set to zero. Gene–reaction associations model the logical relationship between genes and their corresponding reactions and include cases such as isozymes, multifunctional proteins and protein complexes. If a single gene is associated with multiple reactions, the deletion of that gene will result in the removal of all associated reactions. On the other hand, a reaction that can be catalyzed by multiple non-interacting gene products will not be removed in a single gene deletion. The possible results from a simulation of a single or double gene deletion are unchanged maximal growth (non-lethal), reduced maximal growth or no growth (lethal). Whereas it is theoretically possible to attempt double deletion of every possible gene pair experimentally, the sheer number of possible two-gene deletions makes this virtually impossible. However, computational predictions of double gene deletion phenotypes can be made in a matter of hours and the results can be used to guide the design of informative confirmation experiments. Perform a model-wide single gene deletion study using the following function:

```
[grRatio, grRateKO, grRateWT] = singleGeneDeletion(model,method)
```

This function calculates the growth rates for each deletion strain (grRateKO) as well as the relative growth rate ratios (grRatio) between the deletion strains and the wild type (grRateWT). The variable method allows selecting the computational method used to compute the gene deletion results. Specifically, setting method to FBA uses FBA alone to calculate the results, whereas setting method to lMOMA uses a linear version of MOMA[25] to calculate the results (see ANTICIPATED RESULTS). Determine the effects of all pairwise double gene deletions using the function:

```
[grRatio, grRateKO, grRateWT] = doubleGeneDeletion(model,method)
```

This function calculates the growth rates (grRateKO) and relative growth ratios (grRatio) for each possible two-gene combination by a method similar to the single deletion study and outputs each in a matrix with rows and columns corresponding to all genes in model.genes. Using the relative growth rate data, the following function can be utilized to find epistatic (synthetic lethal or synthetic sick) interactions between genes in the model:

```
interactions = findEpistaticInteractions(model,grRatio)
```

By default, this function considers both synthetic lethal and synthetic sick interactions, where a synthetic sick interaction is defined as one where the double deletion strain fitness values (or growth rate ratio) are at least 0.01 smaller than either of the single deletion strain fitness values.

Flux Variability Analysis (FVA)

11| Biological systems often contain redundancies that contribute to their robustness. FVA can be used to examine these redundancies by calculating the full range of numerical values for each reaction flux in a network. This is carried out by optimizing for a particular objective, while still satisfying the given constraints set on the system[26]. One particularly useful application of FVA is to determine the ranges of fluxes that correspond to an optimal solution determined through FBA. The maximum value of the objective function is first computed and this value is used with multiple optimizations to calculate the maximum and minimum flux values through each reaction. FVA can be used to study the entire range of achievable cellular functions as well as the redundancy in optimal phenotypes[26].

Determine the minimum and maximum flux value that each reaction in the model can possess while satisfying the steady-state assumption of FBA and the constraints on the system using the function:

```
[minFlux,maxFlux] = fluxVariability(model,optPercentage)
```

By setting the optPercentage variable (allowable range: 0–100), the analysis will only consider solutions that give you at least a certain percentage of the optimal solution. The default value for the optPercentage variable is 100 (i.e., optimal solution(s) only). minFlux and maxFlux are vectors that contain the calculated minimum and maximum flux value for each reaction in the model using FVA.

Sampling of allowed flux distributions

12| FBA can generate a single flux distribution corresponding to maximal growth under any given environmental condition. An alternative approach would be to characterize the space of all flux distributions that are allowed by the mass balance (stoichiometric) and flux capacity constraints. There are a number of methods that allow characterizing entire flux solution spaces, including various forms of pathway analysis[9], but these methods are generally not scalable to genome-scale models containing thousands of reactions. Uniform random sampling of the solution space in any environmental condition is a rapid and scalable way to characterize the structure of the allowed space of metabolic fluxes. The set of flux distributions obtained from sampling can be interrogated further to answer a number of questions related to the metabolic network function. Such questions include, for example, the most likely flux value through any reaction, given the environmental constraints imposed on the organism and how dependent two reactions within the network are on each other.

Perform uniform random sampling of the flux space defined by the constraints in the model with the function:

```
[modelS,samples] = sampleCbModel(model,sampleFile)
```

The sampleFile variable contains the file name used to store sampling results. If no options variable is provided, the sampling is done using default options as described in the documentation to sampleCbModel. The function returns the actual pre-processed model used for sampling (modelS) as well as a set of 2,000 sampled flux distributions (samples) in a matrix form.

Plot histograms of flux distributions for individual reactions as well as pairwise scatterplots for the reactions of interest with the function:

```
sampleScatterMatrix(rxnNames,modelS,samples)
```

The set of reactions for which sampling information is displayed is defined in rxnNames.

Finding modules in metabolic networks

13| Sampling can be used to determine dependencies between reactions that can be further used to define modules of reactions in networks that have to be co-utilized in precise stoichiometric ratios (correlated reaction sets). Correlated reactions sets are unbiased, condition-dependent definitions of modules in metabolic networks that can, for example, be compared with modules obtained using gene expression data[27].

Starting with a set of samples (the output from first calling sampleCbModel), identify the correlated reactions sets with:

```
[sets,setNumber,setSize] = identifyCorrelSets(modelS,samples,corrThr)
```

The inputs to the function are the model (modelS) and a set of random flux samples (samples) as well as an optional argument defining the minimum correlation threshold (corrThr). The function returns a list of the correlated reaction sets (sets) as well as the set number that each reaction belongs to (vector setNumber) and the set sizes (vector setSize). If a reaction is not part of any correlated reaction set, the corresponding entry in setNumber equals zero.

● TIMING

There are quite significant speed differences between different LP solvers and thus the timing estimates depend on the solver used. For example, the lp_solve solver with default settings is significantly slower than the LINDO, glpk or CPLEX solvers on the type of problems discussed here. The solution of a single linear optimization problem using optimizeCbModel for the yeast model typically takes approximately 1.3 s on a newer personal computer (Intel Core 2 Duo 6600 2.4 GHz with 2 Gb of memory running Windows 2003 Server) using lp_solve, 0.17 s using LINDO API 4.1 and 0.025 s using CPLEX through Tomlab (all with default settings for the solvers). All the other calculations scale in proportion to the number of LP problems that have to be solved. For example, a model-wide single gene deletion study for yeast models takes approximately 750 times the time it takes to solve one LP problem (17 min with lp_solve, 2 min with LINDO and 20 s with CPLEX). FVA requires solving two LPs for each reaction included in the model (1,266 for the yeast model), resulting in run times of 28 min for lp_solve, 4 min for LINDO and 30 s for CPLEX. A double deletion study for yeast that requires the calculation of approximately 210,000 single LP would take 76 h with lp_solve, 10 h with LINDO and 1.5 h with CPLEX. Sampling with the standard settings takes approximately 2 h on a personal computer, but many fewer sample points are generally sufficient for calculating correlated reaction sets.

? TROUBLESHOOTING

Troubleshooting advice can be found in **Table 1**.

TABLE 1 | Troubleshooting table.

Step and problem	Solution
2. SBML file not read correctly	Check that the SBML Toolbox is installed correctly, the SBML Toolbox is accessible in your Matlab path and the test cases included in the SBML Toolbox work as outlined Check that the SBML file you are using contains all the information needed for the COBRA Toolbox (see **Supplementary Information**)
7. Linear programming solver not accessible	Make sure that the installation instructions for the LP solver are followed carefully and test the solver using the examples provided with the solver Check that the solver and the Matlab interface files are accessible in the Matlab path Check that the default solver for the COBRA toolbox is set correctly (initCobraToolbox script or changeCobraSolver function)
7. optimizeCbModel yields an infeasible or infinite solution	Check that the constraints for the model are not conflicting and should allow experimental growth of the cell (i.e., all the necessary nutrients are provided to the model) Check that at least one of the constraints is limiting (e.g., glucose or oxygen uptake rate)

ANTICIPATED RESULTS

Optimal flux distributions and growth rates for *Escherichia coli*

To read in the *E. coli i*JR904 model provided in **Supplementary Information** and on our website and use FBA to predict flux distribution for optimal growth on glucose minimal media, the following functions are called sequentially in Matlab:

```
model = readCbModel('Ec_iJR904_GlcMM');
solution1 = optimizeCbModel(model);
printFluxVector(model, solution1.x, true, true);
```

The expected result for the objective value, solution1.f, is 0.539. The last function prints out the non-zero uptake and secretion fluxes in the optimal flux solution.

The following sequence of commands can be used to simulate the change of carbon source from glucose to succinate (uptake rate set also changed to 9 mmol gDW^{-1} h^{-1}) and to obtain the corresponding optimal growth rates and flux distributions:

```
model2 = changeRxnBounds(model, {'EX_glc(e)','EX_succ(e)'}, [0 -9],'l');
solution2 = optimizeCbModel(model2);
printFluxVector(model2, solution2.x, true, true);
```

The expected result is 0.386.

To simulate anaerobic growth for a glucose minimal medium, the oxygen input must be set to zero before solving the FBA problem, using the following functions:

```
model3 = changeRxnBounds(model,'EX_o2(e)', 0,'l');
solution3 = optimizeCbModel(model3);
printFluxVector(model3, solution3.x, true, true);
```

The expected result is 0.097, showing substantially decreased anaerobic growth as compared with aerobic growth with the same glucose uptake rate. The growth rates corresponding to the three conditions (glucose aerobic, succinate aerobic and glucose anaerobic) are shown in **Table 2**.

TABLE 2 | The expected results from basic growth simulations.

Conditions	Biomass flux (h^{-1})
Glucose aerobic	0.539
Succinate aerobic	0.386
Glucose anaerobic	0.097

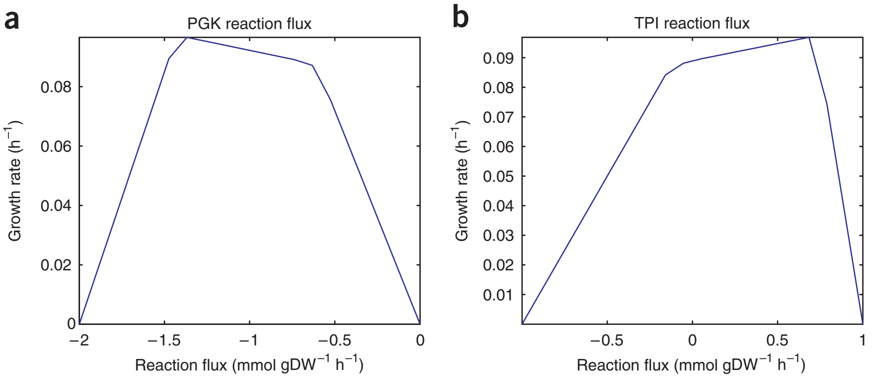


Figure 4 | Robustness analysis. The predicted optimal growth rate of yeast in aerobic, glucose minimal media conditions as a function of the flux through the (**a**) PGK and (**b**) TPI reactions.

Robustness analysis of yeast central metabolism

The sensitivity of the predicted growth rate to changing the flux through glycolytic reactions, *PGK* (phosphoglycerate kinase) and *TPI* (triose phosphate isomerase), is studied here. The following functions were used to perform the analysis and to generate plots that show how optimal growth rates vary as a function of flux through these reactions:

```
model = readCbModel('Sc_iND750_GlcMM');
robustnessAnalysis(model, 'PGK', 20);
robustnessAnalysis(model, 'TPI', 20);
```

The growth rate is sustained near the optimal value over a range of values for both *PGK* and *TPI* (**Fig. 4**), indicating network robustness with respect to flux changes in both reactions. However, a complete deletion of the PGK reaction would be predicted to result in a lethal phenotype, whereas deletion of the TPI reaction is predicted to cause modest growth retardation.

Dynamic growth simulations (batch growth) of *E. coli* on glucose minimal media

Dynamic FBA is performed for the *E. coli* iJR904 model to simulate aerobic batch growth in glucose minimal media conditions. The maximum uptake rates of glucose and oxygen are changed to 10 and 18 mmol gDW^{-1} h^{-1}, respectively, and the initial glucose concentration is set to 10 mM. The initial biomass concentration is set to 0.035 gDW L^{-1}. The time step is 15 min and the maximum number of steps is set to 20 in order to allow observing the full diauxic shift. The Matlab functions to perform batch simulations and to plot the results are:

```
model = readCbModel('Ec_iJR904_GlcMM');
model = changeRxnBounds(model, {'EX_glc(e)','EX_o2(e)'},...
[−10 −18],'l');
substrateRxns = {'EX_glc(e)'};
initConcentrations = 10; initBiomass = .035;
timeStep = .25; nSteps = 20;
plotRxns = {'EX_glc(e)','EX_ac(e)'};
dynamicFBA(model,substrateRxns,initConcentrations,...
initBiomass,timeStep,nSteps,plotRxns);
```

The expected results from this set of functions are a printout of the predicted biomass concentrations for each time step in the simulation and a plot of the biomass, glucose and acetate concentrations per unit time. The expected results from the batch simulation with these initial values are shown in the form of a growth curve and substrate/by-product concentration curves in **Figure 5**.

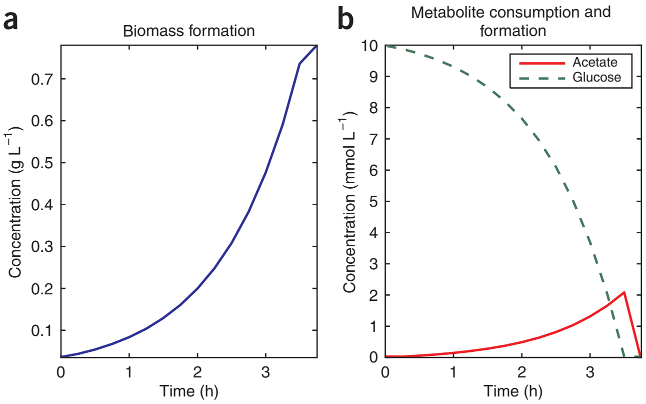


Figure 5 | Dynamic FBA. (**a**) The predicted biomass concentration and (**b**) the glucose and acetate concentrations are shown as a function of time for aerobic *E. coli* batch growth on glucose minimal media.

Single and double gene deletion phenotypes of yeast metabolic genes

To perform a genome-scale single deletion study for yeast on glucose minimal media, the reaction(s) flux corresponding to each deleted gene is set to zero and the relative growth rate of the deleted strain to the wild type is computed. There are multiple alternative computational approaches for determining the deletion strain growth rate. The COBRA Toolbox currently allows using either the standard FBA approach or a linear version of the minimization of metabolic adjustment (linearMOMA) approach. FBA assumes that the deletion strain will still maximize its growth rate, but the missing reaction functionalities may result in a lower optimal growth rate.

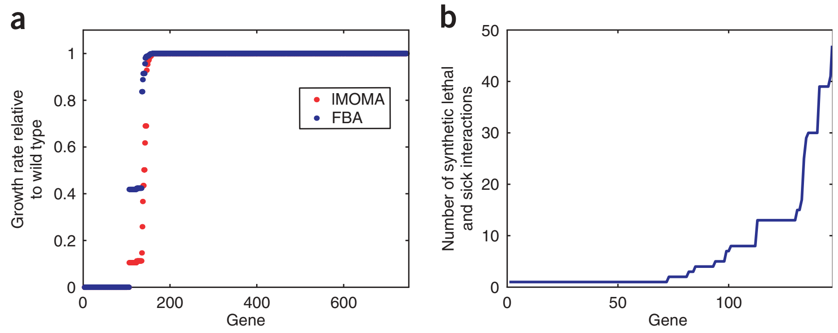


Figure 6 | Single and double deletion predictions using *S. cerevisiae i*ND750. (**a**) Single gene deletion results. The distribution of relative growth rates (mutant/wild type) is shown separately for FBA and linearMOMA calculations. One hundred and five of the 750 model genes were found to be lethal when the gene and corresponding reaction(s) were deleted from the model. (**b**) Double gene deletion results (FBA only). Distribution of the number of synthetic lethal and synthetic sick interactions for each of the 147 genes that participate in at least one such interaction is shown.

In contrast, the linearMOMA approach assumes that the deletion strain minimizes the overall flux adjustment it makes from the wild-type flux distribution. Details on how linearMOMA differs from the published MOMA method are included in **Supplementary Information**. The Matlab functions used to perform single deletions of all the 750 genes included in the model using both the default FBA and linearMOMA methods and plot the results are as follows:

```
model = readCbModel(‘Sc_iND750_GlcMM’);
grRatioFBA = singleGeneDeletion(model,‘FBA’);
grRatioMOMA= singleGeneDeletion(model,‘lMOMA’);
plot(1:length(grRatioFBA), [sort(grRatioMOMA)…
sort(grRatioFBA)],‘.’);
```

The results in **Figure 6a** show the distribution of predicted relative growth rates (grRatio and grRatioMOMA) for all the gene deletions in the *i*ND750 model. Out of the 750 genes in the model, 105 genes were considered lethal and 645 genes were non-lethal, with 57 gene deletions resulting in reduced maximal growth rates and 593 exhibiting no change in growth (defined as mutant growing at $>$99.9% of the wild-type growth rate). In a published *i*ND750 study, 4,154 predicted growth phenotypes across multiple environmental conditions were compared with two large-scale, experimental deletion studies and were found to be in 83% agreement[16].

A genome-scale double deletion study under the same conditions can be performed using the doubleGeneDeletion function:

```
grRatioDouble = doubleGeneDeletion(model,‘FBA’);
```

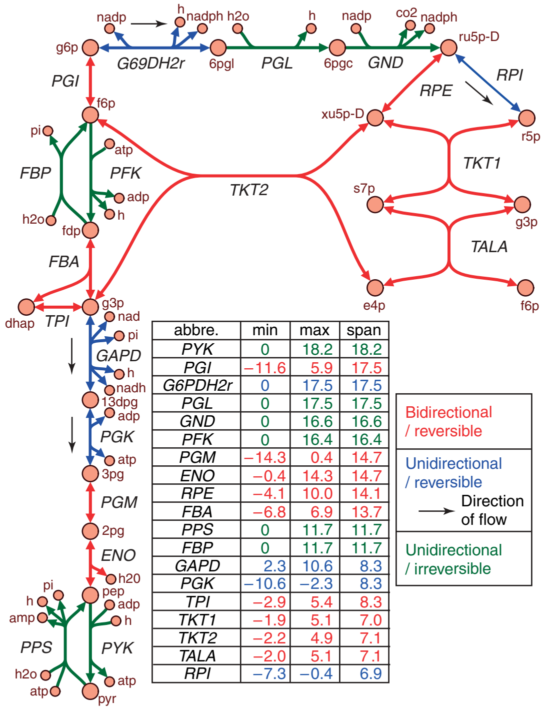
The double deletion fitness values (growth rate of the double deletion strain divided by the wild-type strain growth rate) can then be used to determine epistatic interactions and to plot the distribution of the number of epistatic interactions per gene using the following functions:

```
interactions = findEpistaticInteractions(model,grRatioDouble);
nInteractions = sum(interactions);
plot(sort(nInteractions(nInteractions > 0)),‘-’);
```

The distribution of the number of synthetic lethal or synthetic sick interactions for each of the genes that has at least one such interaction is shown in **Figure 6b**. As the linearMOMA approach is significantly more time consuming than the FBA approach, we only use the FBA approach in the double deletion study.

Flux variability in *E. coli* central metabolic pathways

FVA is performed for the *E. coli* model *i*JR904 under glucose-limited aerobic growth conditions. Cellular growth is constrained to be within 90% of the maximum value



abbre.	min	max	span
PYK	0	18.2	18.2
PGI	−11.6	5.9	17.5
G6PDH2r	0	17.5	17.5
PGL	0	17.5	17.5
GND	0	16.6	16.6
PFK	0	16.4	16.4
PGM	−14.3	0.4	14.7
ENO	−0.4	14.3	14.7
RPE	−4.1	10.0	14.1
FBA	−6.8	6.9	13.7
PPS	0	11.7	11.7
FBP	0	11.7	11.7
GAPD	2.3	10.6	8.3
PGK	−10.6	−2.3	8.3
TPI	−2.9	5.4	8.3
TKT1	−1.9	5.1	7.0
TKT2	−2.2	4.9	7.1
TALA	−2.0	5.1	7.1
RPI	−7.3	−0.4	6.9

Bidirectional / reversible

Unidirectional / reversible

→ Direction of flow

Unidirectional / irreversible

Figure 7 | FVA. Shown is a map of some of the reactions in glycolysis and the pentose phosphate pathway of *E. coli*. Using FVA, the minimum (min) and maximum (max) allowable flux values for each reaction were determined using the *E. coli* model *i*JR904. The values shown in the table correspond to the min and max allowable fluxes for each reaction shown in the map when the predicted growth rate is constrained to 90% of the optimal value under glucose-limiting conditions. The results were further characterized by the direction of predicted flux (bidirectional or unidirectional) computed using FVA. The black arrows in the figure show the predicted unidirectional direction of flux for reversible reactions that can potentially operate in either direction.

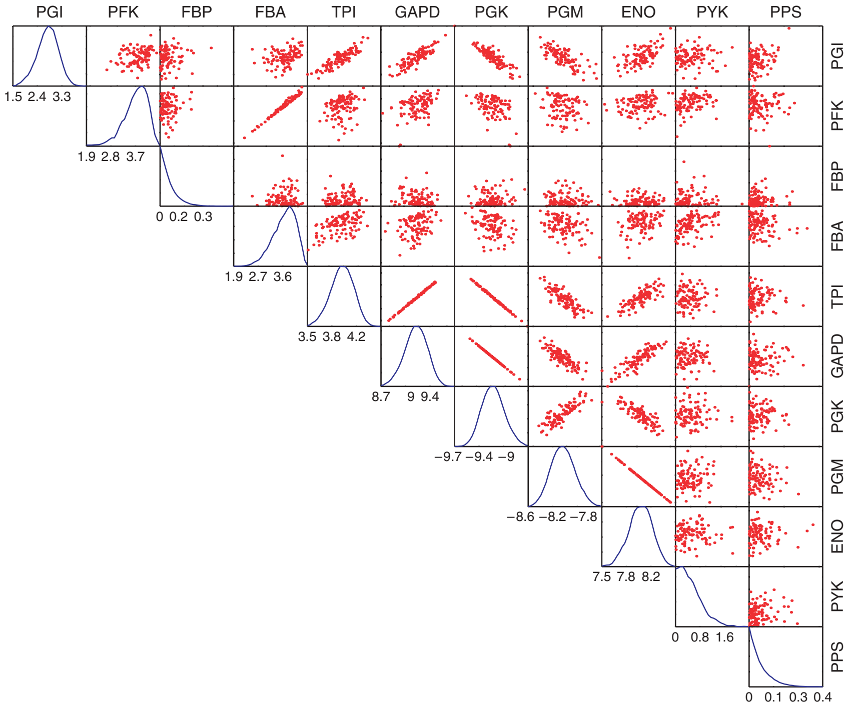


Figure 8 | Flux sampling of *E. coli*. Flux distribution histograms (diagonal) and pairwise scatterplots (off-diagonal) for glycolytic reactions in *E. coli*. Reaction names are the same as in **Figure 7**. The *x* axis of the histograms indicates the magnitude of the flux through the particular reaction. The scatterplots on the off-diagonal elements show the relationship between fluxes through two reactions. For example, TPI and GAPD fluxes are fully correlated in the model, whereas there is very little correlation between TPI and PPS fluxes.

(optPercentage option to fluxVariability). To examine the minimum and maximum fluxes for the reactions in glycolysis and the pentose phosphate pathway, the findRxnIDs function is used to find the indices of the reactions in the model and these indices are then used to select the appropriate entries in the minFlux and maxFlux vectors. The Matlab functions for this analysis are:

```
model = readCbModel(‘Ec_iJR904_GlcMM’);
[minFlux,maxFlux] = fluxVariability(model,90);
rxnNames =...
{‘PGI’,‘PFK’,‘FBP’,‘FBA’,‘TPI’,‘GAPD’,‘PGK’,‘PGM’,‘ENO’,...
‘PYK’,‘PPS’,‘G6PDH2r’,‘PGL’,‘GND’,‘RPI’,‘RPE’,‘TKT1’,...
‘TKT2’,‘TALA’};
rxnID = findRxnIDs(model,rxnNames);
printLabeledData(model.rxns(rxnID),[minFlux(rxnID)...
maxFlux(rxnID) maxFlux(rxnID)-minFlux(rxnID)],true,3);
```

The last function (printLabeledData) is used to print the minimum and maximum allowed fluxes for each reaction, as well as the range of flux values sorted by the range (see **Fig. 7**).

Analyzing flux correlations in *E. coli* glycolysis using sampling

Uniform random sampling is performed for the *E. coli* model *i*JR904 under glucose-limiting aerobic growth conditions. In order to restrict the sampling to solution space relevant to *in vivo E. coli* growth on glucose, a lower bound for the growth rate predicted by the model is set to 90% of the optimal maximum growth rate. Default options are used to sample the model for a total of 10 million steps of the hit-and-run sampler out of which 20,000 flux distributions were saved and 2,000 returned to the user (these parameters can be changed by supplying additional options to sampleCbModel). Histograms and pairwise scatterplots are then displayed for glycolytic reactions (see **Fig. 7** for reaction names). The following functions perform the steps described above:

```
model = readCbModel(‘Ec_iJR904_GlcMM’);
sol = optimizeCbModel(model);
growthRate = sol.f;
model = changeRxnBounds(model,‘BiomassEcoli’,0.9*growthRate,‘l’);
[modelSampling,samples] =...
sampleCbModel(model,‘Ec_iJR904_GlcMM_flux’);
rxnNames = {‘PGI’,‘PFK’,‘FBP’,‘FBA’,‘TPI’,‘GAPD’,‘PGK’,‘PGM’,‘ENO’,‘PYK’,‘PPS’};
sampleScatterMatrix(rxnNames,modelSampling,samples);
```

The output of sampleScatterMatrix is shown in **Figure 8**.

Correlated reaction sets in the *Saccharomyces cerevisiae* metabolic network

After performing uniform random sampling of *S. cerevisiae i*ND750 in aerobic glucose minimal medium, correlated reaction sets are identified using identifyCorrelSets. The five largest correlated reaction sets are selected for further analysis.

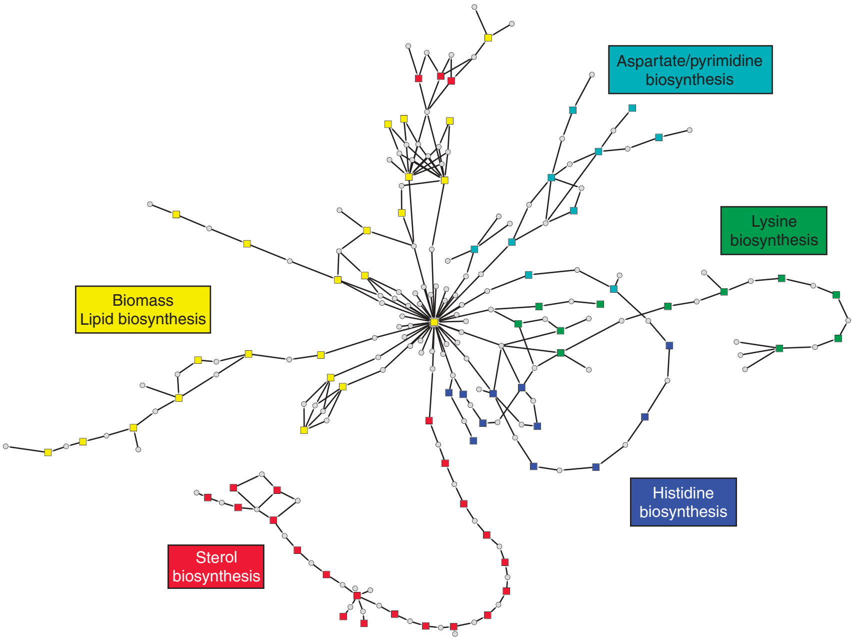


Figure 9 | Correlated reaction sets in yeast. Metabolic subnetworks participating in the five largest correlated reaction sets. Reactions are shown as colored boxes and metabolites as gray circles. The reactions participating in the five correlated reaction sets are indicated by different colors. The reaction in the middle of the plot that connects all five sets is the biomass production reaction. Highly connected metabolites such as cofactors and water have been filtered out before laying out the network. The figure was created using Cytoscape.

The Cytoscape input file of the subnetworks corresponding to the five largest correlated reaction sets was generated using outputNetworkCytoscape. The following functions perform the steps described above:

```
model = readCbModel('Sc_iND750_GlcMM');
[modelSampling,samples] =...
sampleCbModel(model,'Sc_iND750_GlcMM_flux');
[sets,setNumber] =...
identifyCorrelSets(modelSampling,samples);
selectRxns = setNumber > 0 & setNumber <=5;
outputRxnList = model.rxns(selectRxns);
outputSetNumber = setNumber(selectRxns);
outputNetworkCytoscape(model,'iND750_correlSets',...
outputRxnList,outputSetNumber,40);
```

The five subnetworks corresponding to the correlated reaction sets, as well as their connections, are shown in **Figure 9**.

Note: Supplementary information is available via the HTML version of this article.

ACKNOWLEDGMENTS We acknowledge Nathan Price, Vasiliy Portnoy, Jan Schellenberger and Christian Barrett for help with the COBRA Toolbox development and testing. Support for this work was provided by the National Institutes of Health (R01 GM071808, 2R01 GM062791-04A2) and National Science Foundation (BES-0331342).

COMPETING INTERESTS STATEMENT The authors declare competing financial interests (see HTML version of this article for details).

Published online at http://www.natureprotocols.com
Rights and permissions information is available online at http://npg.nature.com/reprintsandpermissions

1. Bork, P. Is there biological research beyond Systems Biology? A comparative analysis of terms. *Mol. Syst. Biol.* **1** Epub 2005 May 25 (2005).
2. Papin, J.A., Hunter, T., Palsson, B.O. & Subramaniam, S. Reconstruction of cellular signalling networks and analysis of their properties. *Nat. Rev. Mol. Cell. Biol.* **6**, 99–111 (2005).
3. Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J. & Palsson, B.O. Integrating high-throughput and computational data elucidates bacterial networks. *Nature* **429**, 92–96 (2004).
4. Brynildsen, M.P., Wong, W.W. & Liao, J.C. Transcriptional regulation and metabolism. *Biochem. Soc. Trans.* **33**, 1423–1426 (2005).
5. Reed, J.L., Famili, I., Thiele, I. & Palsson, B.O. Towards multidimensional genome annotation. *Nat. Rev. Genet.* **7**, 130–141 (2006).
6. Papin, J.A. *et al.* Comparison of network-based pathway analysis methods. *Trends Biotechnol.* **22**, 400–405 (2004).
7. Papin, J.A. & Palsson, B.O. The JAK-STAT signaling network in the human B-cell: an extreme signaling pathway analysis. *Biophys. J.* **87**, 37–46 (2004).
8. Reed, J.L., Vo, T.D., Schilling, C.H. & Palsson, B.O. An expanded genome-scale model of *Escherichia coli* K-12 (iJR904 GSM/GPR). *Genome Biol.* **4**, R54 (2003).
9. Price, N.D., Reed, J.L. & Palsson, B.O. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat. Rev. Microbiol.* **2**, 886–897 (2004).
10. Fong, S.S. & Palsson, B.O. Metabolic gene-deletion strains of *Escherichia coli* evolve to computationally predicted growth phenotypes. *Nat. Genet.* **36**, 1056–1058 (2004).
11. Hong, S.H. *et al.* The genome sequence of the capnophilic rumen bacterium *Mannheimia succiniciproducens*. *Nat. Biotechnol.* **22**, 1275–1281 (2004).
12. David, H., Akesson, M. & Nielsen, J. Reconstruction of the central carbon metabolism of *Aspergillus niger*. *Eur. J. Biochem./FEBS* **270**, 4243–4253 (2003).
13. Sheikh, K., Forster, J. & Nielsen, L.K. Modeling hybridoma cell metabolism using a generic genome-scale metabolic model of *Mus musculus*. *Biotechnol. Prog.* **21**, 112–121 (2005).
14. Kuepfer, L., Sauer, U. & Blank, L.M. Metabolic functions of duplicate genes in *Saccharomyces cerevisiae*. *Genome Res.* **15**, 1421–1430 (2005).
15. Tong, A.H. *et al.* Global mapping of the yeast genetic interaction network. *Science* **303**, 808–813 (2004).
16. Duarte, N.C., Herrgard, M.J. & Palsson, B.O. Reconstruction and validation of *Saccharomyces cerevisiae* iND750, a fully compartmentalized genome-scale metabolic model. *Genome Res.* **14**, 1298–1309 (2004).
17. Fong, S.S. *et al. In silico* design and adaptive evolution of *Escherichia coli* for production of lactic acid. *Biotechnol. Bioeng.* **91**, 643–648 (2005).
18. Wang, Q., Chen, X., Yang, Y. & Zhao, X. Genome-scale *in silico* aided metabolic analysis and flux comparisons of *Escherichia coli* to improve succinate production. *Appl. Microbiol. Biotechnol.* (2006).
19. Alper, H., Jin, Y.S., Moxley, J.F. & Stephanopoulos, G. Identifying gene targets for the metabolic engineering of lycopene biosynthesis in *Escherichia coli*. *Metab. Eng.* **7**, 155–164 (2005).

20. Klamt, S., Stelling, J., Ginkel, M. & Gilles, E.D. FluxAnalyzer: exploring structure, pathways, and flux distributions in metabolic networks on interactive flux maps. *Bioinformatics* **19**, 261–269 (2003).
21. Zamboni, N., Fischer, E. & Sauer, U. FiatFlux—a software for metabolic flux analysis from ^{13}C-glucose experiments. *BMC Bioinformatics* **6**, 209 (2005).
22. Hucka, M. *et al.* The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**, 524–531 (2003).
23. Shannon, P. *et al.* Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504 (2003).
24. Varma, A. & Palsson, B.O. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type *Escherichia coli* W3110. *Appl. Environ. Microbiol.* **60**, 3724–3731 (1994).
25. Segre, D., Vitkup, D. & Church, G.M. Analysis of optimality in natural and perturbed metabolic networks. *Proc. Natl. Acad. Sci. USA* **99**, 15112–15117 (2002).
26. Mahadevan, R. & Schilling, C.H. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metab. Eng.* **5**, 264–276 (2003).
27. Reed, J.L. & Palsson, B.O. Genome-scale *in silico* models of *E. coli* have multiple equivalent phenotypic states: assessment of correlated reaction subsets that comprise network states. *Genome Res.* **14**, 1797–1805 (2004).