1. Estimation of flux through a metabolic network

Our overall objective as a metabolic engineer is to maximize the `performance` of a metabolic network, e.g., develop a system that produces a desired product with the highest possible yield, or at the maximum possible rate, etc.

Toward this goal, we developed material balances that describe the dynamic and steady-state `concentration` of chemical species (metabolites) in a metabolic network. In particular, the `concentration` of metabolites (written in cell mass-specific units, e.g., $\star$mol/gDW) for a metabolic network with $\mathcal{R}$ reactions and $\mathcal{M}$ metabolites operating in batch culture can be written (in matrix-vector form) as:

$$\frac{d\mathbf{x}}{dt} = \mathbf{S}\mathbf{v} - \mu\mathbf{x}$$

where $\mathbf{x}$ denbotes the $\mathcal{M}\times 1$ column vector of metabolite concentrations, $\mathbf{S}$ denotes the $\mathcal{M}\times\mathcal{R}$ stoichiometricx matrix, $\mathbf{v}$ denotes the $\mathcal{R}\times 1$ column vector of (net)reaction rates, and $\mu\mathbf{x}$ denotes the diliution due to growth term. Note: the form of the dilution terms depends upon the concentration basis.

We also introduced some tools that we might use to achieve our engineering objective. In particular, we introduced some tools to explore the structure and properties of the stoichiometric matrix $\mathbf{S}$, and Flux Balance Analysis (FBA), a tool to compute optimal flux through a metabolic network.

Today, we will:

- Compute extreme pathways using the expa algorithm
- Compute optimal `flux` through a metabolic network using flux balance analysis (FBA)

2. Extreme pathways

The set of extreme pathways, a generating set for all possible steady-state flux distributions in a metabolic network, can be computed from the stoichiometric matrix $\mathbf{S}$. In particular, any steady-state flux through a metabolic network can be written as:

$$\mathbf{v} = \sum_{i=1}^{\mathcal{P}} \alpha_i \mathbf{P}_i$$

where $\mathcal{P}$ denotes the number of convex basis pathways, and $\mathbf{P}_i$ denotes the $\mathcal{R} \times 1$ extreme pathway basis vector. The term(s) $\alpha_i \geq 0$ denotes the weight of extreme pathway i.

For more information on extreme pathways, check out:

- Lecture 12: Pathways. Systems Biology: Constraint-based Reconstruction and Analysis, Bernhard Ø. Palsson, Cambridge University Press, 2015

2.1 Computing extreme pathways using expa

We've implemented the expa algorithm from Palsson and coworkers:

- Bell SL, Palsson BØ. Expa: a program for calculating extreme pathways in biochemical reaction networks. Bioinformatics. 2005 Apr 15;21(8):1739-40. doi: 10.1093/bioinformatics/bti228. Epub 2004 Dec 21. PMID: 15613397.

Let's use expa to explore the toy network taken from (note: we've renumbered the reactions):

- Schilling CH, Letscher D, Palsson BO. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. J Theor Biol. 2000 Apr 7;203(3):229-48. doi: 10.1006/jtbi.2000.1073. PMID: 10716907.

```julia
begin

    # Setup a collection of reaction strings -
    reaction_array = Array{String,1}()

    # Setup a collection of reaction strings -
    reaction_array = Array{String,1}()

    # encode the reactions -
    # internal reactions -
    push!(reaction_array,"v₁,A,B,false")
    push!(reaction_array,"v₂,B,C,true")
    push!(reaction_array,"v₃,C,D,true")
    push!(reaction_array,"v₄,C,E,false")

    # exchange reactions -
    push!(reaction_array,"b₁,∅,A,true")
    push!(reaction_array,"b₂,∅,B,true")
    push!(reaction_array,"b₃,D,∅,true")
    push!(reaction_array,"b₄,E,∅,true")

    # compute the stoichiometric matrix -
    # the optional expand arguement = should we split
   reversible reactions? (default: false)
    (S, species_array, reaction_name_array) =
   lib.build_stoichiometric_matrix(reaction_array;
        expand=true);

    # show -
    nothing
end
```

```julia
(ℳ,ℛ) = size(S)
```

```julia
S
```

```julia
begin

    # compute the extreme pathways Tableu -
    PM = lib.expa(S)

    # P constaints the extreme pathways (rows) and 𝒩 is the
   "balanced" array (should be all zeros) -
    P = PM[:,1:ℛ]
    𝒩 = PM[:,(ℛ+1):end]

    # show -
    nothing
end
```

```
size(P)
```

```
rank(P)
```

```
pidx = 7
```

Table 1: Extreme pathways table for the toy network of Schilling CH, Letscher D, Palsson BO. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. J Theor Biol. 2000 Apr 7;203(3):229-48. doi: 10.1006/jtbi.2000.1073. PMID: 10716907.

3. Flux Balance Analysis (FBA)

Flux balance analysis (FBA) is a mathematical modeling and analysis approach that estimates the *intracellular* reaction rate (metabolic flux) of carbon and energy throughout a metabolic network (units: $\star$mol/gDW-time or $\star$mol/L-time for cell-free networks). FBA arguably the most widely used tool in this space. However, there are alternatives to FBA, such as metabolic flux analysis (MFA), but these alternatives vary more in the solution approach than the structure of the estimation problem.

Let's look at the following reference to understand better the different components of a flux balance analysis problem:

- Orth, J., Thiele, I. & Palsson, B. What is flux balance analysis?. Nat Biotechnol 28, 245–248 (2010). https://doi.org/10.1038/nbt.1614

Computational tools for working with flux balance analysis models:

- Heirendt, Laurent et al. "Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v.3.0." Nature protocols vol. 14,3 (2019): 639-702. doi:10.1038/s41596-018-0098-2

3.1 Flux balance analysis problem structure

The FBA problem is typically encoded as a linear programming (LP) problem of the form:

$$\max_{v}\sum_{i=1}^{\mathcal{R}}c_{i}v_{i}$$

subject to the constraints:

$$\begin{aligned}\mathbf{S}\mathbf{v} &= \mathbf{0}\\ \mathcal{L}\leq\mathbf{v} &\leq\mathcal{U}\\ \dots &= \dots\end{aligned}$$

where $\mathbf{S}$ denotes the stoichiometric matrix, c_i denote the objective coefficients, $\mathbf{v}$ denotes the metabolic flux (the unknown that we are trying to estimate), and $\mathcal{L}$ ($\mathcal{U}$) denote the permissible lower (upper) bounds on the *unkown* metabolic flux. The first set of constraints enforces conservation of mass, while the second imparts thermodynamic and kinetic information into the calculation. Finally, there are potentially other types of constraints (both linear and nonlinear) used in this type of problem (we will not cover these here, but these additional constraints may be important in specific applications that we see later).

3.2 Toy FBA example

Let's consider the `toy` network above. Assume this network is operating in cells growing in a continuous culture at a steady-state, where we use cell mass specific units. We'll use the GNU Linear Programming Kit (GLPK) and the GLPK.jl package to solve the linear programming problem.

```
begin

    # species bounds array -
    # Sv = 0
    # Solver by default takes L <= Sv <= U
    species_bounds_array = zeros(ℳ,2);

    # specify the flux_bounds -
    flux_bounds_array = [

        # v₁ (ℒ,𝒰)
        0.0 100.0       ; # 1 v₁ units: mmol/gDW-hr

        # v₂
        0.0 100.0       ; # 2 Fv₂ units: mmol/gDW-L
        0.0 100.0       ; # 3 Rv₂ units: mmol/gDW-L

        # v₃
        0.0 100.0       ; # 4 Fv₃ units: mmol/gDW-L
        0.0 100.0       ; # 5 Rv₃ units: mmol/gDW-L

        # v₄
        0.0 100.0       ; # 6 v₄ units: mmol/gDW-L

        # b₁
        0.0 100.0       ; # 7 Fb₁ units: mmol/gDW-L
        0.0 0.0         ; # 8 Rb₁ units: mmol/gDW-L

        # b₂
        0.0 100.0       ; # 9 Fb₂ units: mmol/gDW-L
        0.0 100.0       ; # 10 Rb₂ units: mmol/gDW-L

        # b₃
        0.0 100.0       ; # 11 Fb₃ units: mmol/gDW-L
        0.0 0.0         ; # 12 Rb₃ units: mmol/gDW-L
```

```
result
```

Table 2: Flux table for maximizing toy network. The GNU Linear Programming Kit solver produced an estimate of metabolic flux ($\star$ mol/gDW-hr) given the problem constraints.

4. Some fun with constraints and the FBA problem formulation

The standard flux balance analysis problem is written in `concentration` units and metabolic reaction flux, e.g., $\star$mol/gDW-hr. However, there is nothing that says we have to do that. For example, it may be more convenient to work in mass or mole units instead of working in concentration units. Let's consider the latter.

We know from our material and energy balance class that the open mole balance around component i in the *logical* control volume is given by:

$$\sum_{s=1}^{\mathcal{S}} d_s \dot{n}_{is} + \sum_{j=1}^{\mathcal{R}} \sigma_{ij} \dot{\epsilon}_j = 0 \qquad i = 1, 2, \dots, \mathcal{M}$$

The first term is the rate of `transport` into and from the control volume (units: mol i/time) from $\mathcal{S}$ possible `streams`; the `transport` terms can be physical or logical where $d_s = 1$ if the stream s enters control volume, $d_s = -1$ is stream s exits the control volume. The second summation denotes the reaction terms, where σ_{ij} denote the stoichiometric coefficient describing the connection between metabolite i and reaction j and $\dot{\epsilon}_j$ denote the open extent of reaction (units: mol/time).

Let's suppose that we have a single logical stream entering (s=1) and exiting (s=2) the control volume. In this case, the open mole balance becomes:

$$\dot{n}_{i,2} = \dot{n}_{i,1} + \sum_{j=1}^{\mathcal{R}} \sigma_{ij} \dot{\epsilon}_j \qquad i = 1, 2, \dots, \mathcal{M}$$

These balances can be used as constraints to find the optimal open extent of reaction. In particular, we know that we actually pass $\alpha \leq S\dot{\epsilon} \leq \beta$ to the solve. Thus, because $\dot{n}_{i,2} \geq 0$, the FBA problem is subject to the mol constraints:

$$\dot{n}_{i,1}+\sum_{j=1}^{\mathcal{R}}\sigma_{ij}\dot{\epsilon}_j\geq 0\qquad\forall i$$

In other words, when searching for the optimal set of $\dot{\epsilon}_j$ we have to select values that give physically realistic answers (we can't have a negative mol flow rate). Next, the $\dot{\epsilon}_j$ terms (just flux) are bounded from above and below:

$$\mathcal{L}_j\leq\dot{\epsilon}_j\leq\mathcal{U}_j\qquad j=1,2\dots,\mathcal{R}$$

where the $\mathcal{L}_j$ and $\mathcal{U}_j$ denote the lower and upper bounds that $\dot{\epsilon}_j$ can take, remember that the open extents $\dot{\epsilon}_j$ are just reaction rates times the volume. Thus, the lower and upper bounds describe the permissible range we expect the rate *could* obtain.

Putting everything together gives a slightly different problem formulation to compute the mol/time flux through a reaction network. An objective:

$$\mathcal{O}=\sum_{j=1}^{\mathcal{R}}c_j\dot{\epsilon}_j$$

is minimized (or maximized) subject to a collection of linear constraints:

$$\begin{aligned}\sum_{j=1}^{\mathcal{R}}\sigma_{ij}\dot{\epsilon}_j &\geq -\dot{n}_{i,1}\qquad\forall i\\ \dot{n}_{i,2} &\geq 0\qquad\forall i\\ \mathcal{L}_j\leq\dot{\epsilon}_j\leq\mathcal{U}_j &\qquad j=1,2\dots,\mathcal{R}\end{aligned}$$

4.1 Example: flux balance analysis with mole flow rates and reaction open extent

Let's suppose the objective was to maximize the production of P_x. Let's set up a flux balance analysis problem to solve this problem.

Table 3: Flux table for maximizing P production. The GNU Linear Programming Kit solver produced an estimate of the (open) reaction extents $\dot{\epsilon}_j \ \forall j$ given the problem constraints.

```
begin

    # setup flow rate in
    # units: *mol/time
    n_dot_in = [
        10.0    ; # 1 A₁
        10.0    ; # 2 A₂
        0.0     ; # 3 B
        0.0     ; # 4 P
        0.0     ; # 5 C
        0.0     ; # 6 x
        0.0     ; # 7 y
    ]

    # Setup STM -
    # notice: the STM does NOT have the exchange fluxes b*
    S₂ = [

        # r₁ r₂ r₃
        -1.0 0.0 0.0 ; # 1 A₁
        0.0 0.0 -1.0 ; # 2 A₂
        1.0 -1.0 0.0 ; # 3 B
        0.0 1.0 0.0  ; # 4 P
        0.0 0.0 1.0  ; # 5 C
        -1.0 0.0 1.0 ; # 6 x
        1.0 0.0 -1.0 ; # 7 y
    ];

    # set the flux bounds array -
    # assume: all reactions are *irreversible* -
    flux_bounds_array₂ = [

        # ℒ 𝒰
        0.0 1000.0  ; # 1 r₁
        0.0 1000.0      ; # 2 r₂
        0.0 20.0    ; # 3 r₃
```

Table 4: State table for maximizing P production. The flux balance analysis problem produces estimates for the optimal value of the (open) reaction extents $\dot{\epsilon}_j \; \forall j$ (units: $\star$mol/hr). We can then use the steady-state species mol balances to compute the output composition (stream 1 input, stream 2 is an output)

5. Summary and conclusions

In this lecture we:

- Computed extreme pathways using the expa algorithm
- Computed the optimal `flux` through a metabolic network using flux balance analysis (FBA)

6. Next time:

- Compare a dynamic experiment with a flux balance analysis computation: Varma A, Palsson BO. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia coli W3110. Appl Environ Microbiol. 1994 Oct;60(10):3724-31. doi: 10.1128/aem.60.10.3724-3731.1994. PMID: 7986045; PMCID: PMC201879
- Flux problem basics: how do we compute the bounds of a reaction?

```
TableOfContents(title="📚 Table of Contents", indent=true,
depth=5, aside=true)
```

```
begin

    # import some packages -
    using PlutoUI
    using LinearAlgebra
    using RowEchelon
    using IterativeSolvers
    using Combinatorics
    using Plots
    using GLPK
    using PrettyTables

    # setup paths -
    const _PATH_TO_NOTEBOOK = pwd()
    const _PATH_TO_DATA = joinpath(_PATH_TO_NOTEBOOK,"data")
    const _PATH_TO_FIGS = joinpath(_PATH_TO_NOTEBOOK,"figs")
    const _PATH_TO_SRC = joinpath(_PATH_TO_NOTEBOOK,"src")

    # load the class lib as LectureLib -
    lib = ingredients(joinpath(_PATH_TO_SRC, "Include.jl"));

    # return -
    nothing
end
```

```julia
function ingredients(path::String)

    # this is from the Julia source code (evalfile in
    base/loading.jl)
    # but with the modification that it returns the module
    instead of the last object
    name = Symbol("lib")
    m = Module(name)
    Core.eval(m,
        Expr(:toplevel,
             :(eval(x) = $(Expr(:core, :eval))($name, x)),
             :(include(x) = $(Expr(:top, :include))($name,
x)),
             :(include(mapexpr::Function, x) = $(Expr(:top,
:include))(mapexpr, $name, x)),
             :(include($path))))
    m
end
```

```
html"""
<script>
    // initialize -
    var section = 0;
    var subsection = 0;
    var subsubsection = 0;
    var headers = document.querySelectorAll('h3, h5, h6');

    // main loop -
    for (var i=0; i < headers.length; i++) {

        var header = headers[i];
        var text = header.innerText;
        var original = header.getAttribute("text-original");
        if (original === null) {

            // Save original header text
            header.setAttribute("text-original", text);
        } else {

            // Replace with original text before adding
           section number
            text = header.getAttribute("text-original");
        }

        var numbering = "";
        switch (header.tagName) {
            case 'H3':
                section += 1;
                numbering = section + ".";
                subsection = 0;
                break;
            case 'H5':
                subsection += 1;
                numbering = section + "." + subsection;
                break;
            case 'H6':
```