Summary of Modelling Approaches

Mark Poolman

July 3, 2018

Tuesday L8

Null Space

Right Null Space - Relationships between steady-state reaction fluxes.

Dead reactions Carry no flux at steady-state

Enzyme subsets Reactions carrying flux in fixed ratio.

Elementary modes independent routes through a system

- Balance all metabolites
- Respect irreversibility criteria
- Are minimal (no reaction can be removed).

Null Space

Left Null Space - Relationships between metabolite concentrations.

Conserved Moieties Sets of metabolites whose *sum* of concentrations is constant.

Other uses Not covered on this course.

Linear Programming

Identifies flux patterns with specific properties:

$$\begin{aligned} \min/\max \quad & : \quad \mathbf{v}_{\text{targs}} && \longleftarrow \text{objective} \\ \text{subject to} \quad & \begin{cases} \mathbf{Nv} = \mathbf{0} & \longleftarrow \text{steady state} \\ \max_i \geq \mathbf{v}_i \geq \min_i & \longleftarrow \text{flux constraints} \end{cases} \end{aligned}$$

Typical Objectives:

- Maximise output(s) (need to fix input(s))
- FBA - maximise growth rate for fixed input.
- Minimise input(s) (need to fix output(s))
- Minimise *all* reactions (need to fix input(s) and/or output(s))

Linear Programming

Identifies flux patterns with specific properties:

$$\begin{array}{lll} \min/\max & : & \mathbf{v}_{\text{targs}} \quad \longleftarrow \text{objective} \\ \text{subject to} & \begin{cases} \mathbf{Nv} = \mathbf{0} & \longleftarrow \text{steady state} \\ \max_i \geq \mathbf{v}_i \geq \min_i & \longleftarrow \text{flux constraints} \end{cases} \end{array}$$

Typical Objectives:

- Maximise output(s) (need to fix input(s))
- FBA - maximise growth rate for fixed input.
- Minimise input(s) (need to fix output(s))
- Minimise *all* reactions (need to fix input(s) and/or output(s))

Linear Programming

Identifies flux patterns with specific properties:

$$\begin{aligned} \min/\max \quad & : \quad \mathbf{v}_{\text{targs}} && \longleftarrow \text{objective} \\ \text{subject to} \quad & \begin{cases} \mathbf{Nv} = \mathbf{0} & \longleftarrow \text{steady state} \\ \max_i \geq \mathbf{v}_i \geq \min_i & \longleftarrow \text{flux constraints} \end{cases} \end{aligned}$$

Typical Objectives:

- Maximise output(s) (need to fix input(s))
- FBA - maximise growth rate for fixed input.
- Minimise input(s) (need to fix output(s))
- Minimise *all* reactions (need to fix input(s) and/or output(s))

Linear Programming

Identifies flux patterns with specific properties:

$$\begin{array}{lll} \min/\max & : & \mathbf{v}_{\text{targs}} \longleftarrow \text{objective} \\ \text{subject to} & \begin{cases} \mathbf{Nv} = \mathbf{0} & \longleftarrow \text{steady state} \\ \max_i \geq \mathbf{v}_i \geq \min_i & \longleftarrow \text{flux constraints} \end{cases} \end{array}$$

Typical Objectives:

- Maximise output(s) (need to fix input(s))
- FBA - maximise growth rate for fixed input.
- Minimise input(s) (need to fix output(s))
- Minimise *all* reactions (need to fix input(s) and/or output(s))

Linear Programming

Identifies flux patterns with specific properties:

$$\begin{array}{lll} \min/\max & : & \mathbf{v}_{\text{targs}} \quad \longleftarrow \text{objective} \\ \text{subject to} & \begin{cases} \mathbf{Nv} = \mathbf{0} & \longleftarrow \text{steady state} \\ \max_i \geq \mathbf{v}_i \geq \min_i & \longleftarrow \text{flux constraints} \end{cases} \end{array}$$

Typical Objectives:

- Maximise output(s) (need to fix input(s))
- FBA - maximise growth rate for fixed input.
- Minimise input(s) (need to fix output(s))
- Minimise *all* reactions (need to fix input(s) and/or output(s))

Linear programming in ScrumPy

Create an LP object `lp = m.GetLP()`

Set a constraint `lp.SetFixedFlux({"Ethanol_tx":-1}`

Set an objective `lp.SetObjective(["Glucose_tx“]`

Solve the problem `lp.Solve()`

Obtain the solution `sol = lp.GetPrimSol()`

Linear programming in ScrumPy

Reactions can also be given upper and lower bounds:

lp.SetFluxBound({Reaction:(lower,upper)}

lower=None No lower bound set - 0 for irreversible reactions

upper=None No upper bound set.

Transporter Conventions

- Suffix '_tx' is always used to denote a transporter
- Negative transporter flux always denotes loss from the system (export)
- Positive transporter flux always denotes uptake into the system (import)

Modular Models in ScrumPy

The *Include* directive:

```
Include(
    AutoColi.spy,
        # Automatically generated from Ecocyc
    ETC.spy,
        # The electron transport chain
    Transport.spy,
        # Transport reactions
    AlkaneSynth.spy
        # Exogenous reactions to allow for alkane
)
```