Computational Representation of Metabolic Networks

Mark Poolman

December 8, 2024

End of Part 1

We have now covered enough fundamentals to think about how to use it for modelling.

What do we want to represent and act upon ?

What is ScrumPy ?

A collection of modules (a **package**) providing the ability to define and analyse models.

Everything revolves around the use of model objects:

```
>>> m = ScrumPy.Model("FileName.spy")
```

Where "FileName.spy" is the name of file describing the model.

The ".spy" extension is conventional and convenient, but not mandatory.

and "m" is the model object. In these talks, "m" will always be used to denote the model.

Model Definition

In ScrumPy, a model is defined by one or more text files, defining:

Comments	Ignored by ScrumPy, but are useful to the human reader.
Directives	Not part of the model *per se*, but specify how the model is to be read.
Reactions	Define the metabolic network.
Initialisations	Define parameter values and initial metabolite concentrations (only in kinetic models)

Model Definition

```
# comment, everything from #
# to the end of the line is ignored

Structural()
# a Directive. Do not do any kinetic processing.

Rubisco:                                  # a reaction name
    x_CO2 + RuBP_ch -> 2 PGA_ch           # stoichiometry
    ~                                     # default kinetic

PGK:
    PGA_ch + ATP_ch <> BPGA_ch + ADP_ch
    ~

G3Pdh:
    BPGA_ch + x_NADPH_ch + x_Proton_ch <>
                          x_NADP_ch + GAP_ch + Pi_ch
    ~
```

Model Definition - identifiers

Identifiers = Names

Either:
Any sequence of alphanumeric characters and _ (underscore), not starting with a number e.g.
Valid:

```
Fructose6_Phosphate
AlphaAnaline
```

Invalid:

```
2,3-bisphosphoglycerate
TRANS-23-DEHYDROADIPYL-COA
```

Or:
Any quoted (") sequence of characters.

```
"Saturated-Fatty-Acyl-CoA"
"3-oxo-cis-vaccenoyl-ACPs"
```

Analysing models - the Stoichiometry Matrices

Accessed as m.sm (internal) and m.smx (external):

```
>>> print(m.sm)
```

```
             Ru5Pk   Aldo2   TPT_DHAP   Light_react   TKL
RuBP_ch        1       0        0            0         0
ATP_ch        -1       0        0            1         0
ADP_ch         1       0        0           -1         0
GAP_ch         0       0        0            0        -1
Pi_ch          0       0        1           -1         0
DHAP_ch        0      -1       -1            0         0
F6P_ch         0       0        0            0        -1
E4P_ch         0      -1        0            0         1
X5P_ch         0       0        0            0         1
SBP_ch         0       1        0            0         0
Ru5P_ch       -1       0        0            0         0
```

Analysing models - the Stoichiometry Matrices

Accessed as m.sm (internal) and m.smx (external):

```
>>> print(m.sm)

             Ru5Pk  Aldo2  TPT_DHAP  Light_react  TKL
RuBP_ch        1      0       0          0         0
ATP_ch        -1      0       0          1         0
ADP_ch         1      0       0         -1         0
GAP_ch         0      0       0          0        -1
Pi_ch          0      0       1         -1         0
DHAP_ch        0     -1      -1          0         0
F6P_ch         0      0       0          0        -1
E4P_ch         0     -1       0          0         1
X5P_ch         0      0       0          0         1
SBP_ch         0      1       0          0         0
Ru5P_ch       -1      0       0          0         0
```

Analysing models - the Stoichiometry Matrices

- Stoichiometry matrices behave as a list of rows:

```
>>> print m.sm[0]
[mpq(1,1), mpq(0,1), mpq(0,1), mpq(0,1), mpq(0,1), ...]
```

- Or as a dictionary of rows:

```
>>> print m.sm["RuBP_ch"]
[mpq(1,1), mpq(0,1), mpq(0,1), mpq(0,1), mpq(0,1), ...]
```

- Individual elements can be accessed as `matrix[row,col]`:

```
>>> print m.sm[0,0]
1
>>> print m.sm["RuBP_ch","Ru5Pk"]
1
```

Analysing models - the Stoichiometry Matrices

The null-space is obtained the matrix.NullSpace() method:

```
>>> k = m.sm.NullSpace()
>>> print k
```

```
               c_0  c_1  c_2  c_3  c_4
 Ru5Pk          0    0    0    0   -3
 Aldo2          0    0    0    0   -1
 TPT_DHAP       0    2    1    1   -1
 Light_react   -1   -1    0    1   -9
 TKL            0    0    0    0   -1
 G3Pdh          0    0    0    1   -6
 PGK            0    0    0    1   -6
 TPI            0    1    1    1   -3
 TKL2           0    0    0    0   -1
                     .
                     .
                     .
```

And now

We have covered enough to start the practical.