Practical 5: Determination of minimal media requirements for *Campylobacter jejuni*

Here, we will investigate the genome-scale metabolic model of *Campylobacter jejuni* M1cam to identify minimal media requirements as described in the earlier lecture session. For further details you can refer to Tejera *et al* (2020) Genome-Scale Metabolic Model Driven Design of a Defined Medium for *Campylobacter jejuni* M1cam. *Front. Microbiol.* 11:1072. doi: 10.3389/fmicb.2020.01072

Before you start, read the documentation of the `ScrumPy` LP module (https://mudshark.brookes.ac.uk/ScrumPy/Doc/LinProg). It would be a good idea to open this page in a separate tab so that you can refer to it as you work on the practical. (Note that the examples in the documentation are **not for the same model** that you will be using; they are generic examples.)

1. Download the archive containing the model and extract the files.

a) Start ScrumPy in the folder containing the .spy files and load the top-level model file Campy.spy to create a model object. Note that the model is created in a modular fashion, and the top-level file will load the different components of the model and open a window for each.

b) Examine how the model is created. How many modules are loaded? Can you tell what each module contains? Hint: see section 2.1 of Tejera *et al* 2020.

c) How many reactions and metabolites are present in the model?

d) How many media transporters are present in the model? Hint: all media transporters have suffix “_mm_tx”.

e) How many biomass transporters are present in the model? Hint: all biomass transporters have suffix “_bm_tx”.

2. Set up and solve an LP problem where the objective is to minimise total flux, while producing 1 unit flux of AT

a) Create the LP object with the Campylobacter model as argument as shown in the first steps of the documentation. Hint: lp = m.GetLP()

b) Since the optimisation direction is minimisation, and this is set by default, you don't need to change the direction, though the documentation explains how to do this.

c) Next, set the objective as total flux using the SetObjective function of the LP object. Hint: lp.SetObjective(m.sm.cnames)

d) Constrain the flux of the ATPASE-RXN to 1 using the `SetFixedFlux` function of the LP object. Hint: lp.SetFixedFlux({‘ATPASE-RXN ’:1.0})

e) Solve the LP. (ie lp.Solve()) The message 'Optimal solution' should appear. To obtain the solution use the LP method `GetPrimSol()`. This method returns a dictionary object of reactions in the solution as keys and flux values as values, so for convenience assign a name to this solution, e.g. sol = lp.GetPrimSol(). Examine the solution as follows:

```
for k in sol.keys():
    print (k, sol[k])
```

f) How many reactions are present in the solution? What is the objective value, i.e. the sum of total flux producing 1 unit flux of ATP? Hint: lp.GetObjVal()

g) What are the sources of energy/carbon in your LP solution? Calculate ATP generated per unit carbon uptake. Is oxygen used as terminal electron acceptor in your solution? Hint: Oxygen transporter is named as “O2_tx”

h) Oxygen is not essential for energy production in Campylobacter jejuni. Examine if your model is able to generate 1 flux unit of ATP in the absence of oxygen (constrain the flux in “O2_tx” to zero). If so, what is the terminal electron acceptor used by the model? Calculate ATP generated per unit carbon uptake under anaerobic condition.

3. Now we will add biomass constraint to our LP. Set up and solve an LP problem where the objective is to minimise total flux, while producing biomass components in experimentally observed proportion as defined in FluxDic.py (present in Tools directory).

a) Generate a new LP object with the Campylobacter model

b) Optimisation direction and the SetObjective function remains same as above problem (see 2b and 2c respectively

c) Constrain the flux of the ATPASE-RXN to 16 (cell maintenance cost) using the `SetFixedFlux` function of the LP object.

d) Constrain the flux of the biomass transporters as defined in FluxDic.py using the `SetFixedFlux` function of the LP object. Hint:

```
import FluxDic

fd = FluxDic.fd

lp.SetFixedFlux(fd)
```

e) Solve the LP. (ie lp.Solve()) The message 'Optimal solution' should appear. To obtain the solution use the LP method `GetPrimSol()`. This method returns a dictionary object of reactions in the solution as keys and flux values as values, so for convenience assign a different name to this solution.

f) How many reactions are present in the solution? What is the objective value, i.e. the sum of total flux producing biomass? What is the flux on oxygen import?

g) Note that so far we have not set any constraint on oxygen import therefore, simulating aerobic condition. Now represent microaerophilic condition by constraining the maximum oxygen transport flux to 5.0 (make use of SetFluxBounds() function) and repeat the analysis above. Examine the change in oxygen consumption.

h) How many substrates are imported from the media?

4. We will now use the LP generated above to identify auxotrophic substrates.

a) Solve the LP generated above by blocking uptake (constrain the flux to zero) of individual substrate from the medium. If a solution is obtained implies the blocked substrate is not

auxotrophic and the vice versa. Hint: make use of for loops. Also remember to clear the set constraint (using ClearFluxConstraint() function) on each iteration

```
mediatx = list(filter(lambda s:"_mm_tx" in s, m.sm.cnames))
for r in mediatx:
    lp.SetFixedFlux({r:0})
    lp.Solve(False)
    if lp.IsStatusOptimal() == False:
        print (r)
    lp.ClearFluxConstraint(r)
```

b) What are the auxotrophic substrates?