Notebook for metabolic pathway prediction

Vidianos Giannitsis

January 20, 2024

Contents

1 Description of the core concepts

This notebook is about a personal project of mine, which is mostly done as a learning experience in Julia, but if succesful can have application in my thesis (hence why it is in this directory). The idea is that since we know from literature every pathway a mixed culture fermentation can follow, if we have data for the input and output of either a continuous (in steady state) or batch reactor, we can try to find which pathways were followed and to what extent in each case.

This example can be made into an optimization problem as the extent to which each pathway is followed can be considered the parameters of the simulator and with an L2 loss between experimental data and the output of the simulator, we can optimize it with the classic SciML toolchain.

This can be very useful both for understanding the behaviour of a mixed culture of microorganisms and how they behave in different conditions but also can have very large application in modelling. Modelling mixed cultures is in general fairly hard because of the large amount of processes that can happen, but if we can quantify to what extent each process happens, it makes modelling much easier.

2 Primitives

2.1 Molar mass definition

Since we generally measure concentration in g/l, but reactions are described in molar terms, a very important primitive to implement is molar mass. We define molar mass for a general $\mathrm{C_aH_bO_cN_dS_e}$ compound with the function

```
# Primitive to calculate molar mass
function molar_mass(; C=0, H=0, O=0, N=0, S=0)
mass = 12C + H + 16O + 14N + 32S
end
```

2.2 Molar mass of substances used

and then calculate molar mass for all substances used in the system.

```
# Calculate the molar masses of all used substances
function m_glucose()
molar_mass(C=6, H=12, O=6)
end

function m_fructose()
molar_mass(C=6, H=12, O=6)
end

function m_sucrose()
molar_mass(C=12, H=24, O=12)
end

function m_pyruvate()
molar_mass(C=3, H=4, O=3)
end

function m_hydrogen()
molar_mass(H=2)
```

```
end

function m_oxygen()
    molar_mass(O=2)
end

function m_co2()
molar_mass(C=1, O=2)
end

function m_water()
molar_mass(H=2, O=1)
end

function m_acetate()
molar_mass(C=2, H=4, O=2)
end

function m_propionate()
molar_mass(C=3, H=6, O=2)
end

function m_butyrate()
molar_mass(C=4, H=8, O=2)
end

function m_ethanol()
molar_mass(C=2, H=6, O=1)
end

function m_lactate()
molar_mass(C=3, H=6, O=3)
end

function m_succinate()
molar_mass(C=4, H=6, O=4)
end

function m_formate()
molar_mass(C=1, H=2, O=2)
end

function m_acetaldehyde()
molar_mass(C=2, H=4, O=1)
end

function m_acetone()
molar_mass(C=3, H=6, O=1)
end
```

```
function m_butanol()
molar_mass(C=4, H=10, O=1)
end

function m_valerate()
molar_mass(C=5, H=10, O=2)
end
```

2.3 Concentration to mass

Since we define molar mass, we can easily convert moles to mass and opposite. However, what we typically measure is concentration, so we also need a function to convert mass to concentration, which is easy as concentration to mass is multiplication with volume and the opposite is division. This is shown below

```
function conc_to_mass(st, volume)
new_st = NamedTuple{keys(st)}(values(st).*volume)
end

function mass_to_conc(st, volume)
new_st = NamedTuple{keys(st)}(values(st)./volume)
end
```

3 Core pathways

Then, we can start writing down the metabolic pathways which can happen in this system. The concept is that they all operate in a variable named st (state), which is a named tuple holding the concentration of each compound and return a new state of how the concentrations changed due to this process. Furthermore, they all have one (or multiple) goals, which describe to what extent each reaction is followed.

3.1 Initial state

Therefore, we first need an initial state. A test state used for a lot of what is implemented here is displayed below.

```
state = (glucose = 16.0, pyruvate = 0.0, hydrogen = 0.0, water = 700.0,
 ↪  co2 = 0.0,
        acetate = 0.0, propionate = 0.0, butyrate = 0.0, ethanol = 0.0,
        lactate = 0.0, succinate = 0.0, formate = 0.0, acetaldehyde = 0.0,
        fructose = 0.0, sucrose = 0.0, butanol = 0.0, acetone = 0.0,
        valerate = 0.0, oxygen = 0.0)
```

3.2 Glycolysis definition and explanation

After that, we can start writing down reactions. The first reaction we define is glycolysis, the pathway through which glucose is converted to pyruvate, hydrogen and energy.

```
function glycolysis(st; goal = (; glucose = 0.0))
stoic = (glucose = -1, pyruvate = +2, hydrogen = +2)
mass_stoic = (glucose = stoic.glucose*m_glucose(),
                pyruvate = stoic.pyruvate*m_pyruvate(),
                hydrogen = stoic.hydrogen*m_hydrogen())
goal.glucose <= st.glucose || error("Glucose is not sufficient for this
↪  goal")
change = (goal.glucose - st.glucose)/mass_stoic.glucose
new_st = merge(st,
                (glucose = goal.glucose,
                pyruvate = st.pyruvate + change*mass_stoic.pyruvate,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end
```

The logic of the function is we define the stoichiometry, which is known, convert it to mass stoichiometry with the molar mass primitives defined above, find the factor `change` which calculates the conversion of the reaction in mass terms, from the goal given and update the state so that all compounds are changed by this variable times the mass stoichiometric coefficient. For the variable for which goal is defined, its value can more simply be the value of goal. It also runs an error check if the goal of glucose is larger than the glucose in the initial state. Since it is consumed, it cannot be more than its initial value, so the function should give an error if this is given. The logic of all other core reactions is the same, so it won't be explained again below.

3.3 Other sugars

However, in a lot of cases we don't have only glucose. The case study I am doing contains sucrose and fructose, but other sugars could be similarly defined. Sucrose is hydrolyzed to an equimolar mixture of glucose and fructose, while fructose enters the EMP pathway (glycolysis) producing glyceraldehyde-3-phosphate, which is an intermediate of pyruvate. Since this system tries to look at a bigger picture and not every intermediate of the process, the implementation of fructolysis will be that fructose isomerises to glucose and goes through glycolysis. Theoretically it is not correct, but with the amount of abstracted detail we have assumed, it does not give any error. Below are there implementations.

```
function sucrose_hydrolysis(st; goal = (; sucrose = 0.0))
stoic = (sucrose = -1, glucose = +1, fructose = +1)
mass_stoic = (sucrose = stoic.sucrose*m_sucrose(),
                glucose = stoic.glucose*m_glucose(),
                fructose = stoic.fructose*m_fructose())
goal.sucrose <= st.sucrose || error("Sucrose is not sufficient for this
↪  goal")
change = (goal.sucrose - st.sucrose)/mass_stoic.sucrose
```

```
new_st = merge(st,
                (sucrose = goal.sucrose,
                glucose = st.glucose + change*mass_stoic.glucose,
                fructose = st.fructose + change*mass_stoic.fructose))
end

function fructolysis(st; goal = (; fructose = 0.0))
stoic = (fructose = -1, glucose = +1)
mass_stoic = (fructose = stoic.fructose*m_fructose(),
                glucose = stoic.glucose*m_glucose())
goal.fructose <= st.fructose || error("Fructose is not sufficient for this
↪  goal")
change = (goal.fructose - st.fructose)/mass_stoic.fructose
fruc_st = merge(st,
                (fructose = goal.fructose,
                glucose = st.glucose + change*mass_stoic.glucose))
new_st = glycolysis(fruc_st, goal = (; glucose = st.glucose))
end
```

4 Pathways of pyruvate consumption

As mentioned, pyruvate is the core intermediate of the process, produced during glycolysis. There are many pathways it can partake in, producing different products depending on conditions. The core ones (abstracting intermediates of the processes) are:

- Pyruvate + Water -> Acetate + CO2 + H2
- Pyruvate -> Acetaldehyde + CO2
- 2Pyruvate -> Butyrate + 2CO2
- Pyruvate + H2 -> Lactate
- Pyruvate + CO2 + H2 -> Succinate
- 2Pyruvate + 2H_2 -> Water + Butanol + 2CO_2
- 2Pyruvate + Water -> 3CO2 + 2H_2 + Acetone

```
function pyruv_to_acetate(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -1, water = -1, acetate= +1, hydrogen = +1, co2=+1)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                water = stoic.water*m_water(),
                acetate = stoic.acetate*m_acetate(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                co2 = stoic.co2*m_co2())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪  goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
```

```
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                water = st.water + change*mass_stoic.water,
                acetate = st.acetate + change*mass_stoic.acetate,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                co2 = st.co2 + change*mass_stoic.co2))
end

function pyruv_to_acetaldehyde(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -1, acetaldehyde = +1, co2 = +1)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                acetaldehyde = stoic.acetaldehyde*m_acetaldehyde(),
                co2 = stoic.co2*m_co2())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪  goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                acetaldehyde = st.acetaldehyde +
                ↪  change*mass_stoic.acetaldehyde,
                co2 = st.co2 + change*mass_stoic.co2))
end


function pyruv_to_butyr(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -2, butyrate = +1, co2 = +2)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                butyrate = stoic.butyrate*m_butyrate(),
                co2 = stoic.co2*m_co2())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪  goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                butyrate = st.butyrate + change*mass_stoic.butyrate,
                co2 = st.co2 + change*mass_stoic.co2))
end

function pyruv_to_butanol(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -2, hydrogen = -2, water = +1, butanol = +1, co2 = +2)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                water = stoic.water*m_water(),
                butanol = stoic.butanol*m_butanol(),
                co2 = stoic.co2*m_co2())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪  goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
```

```
abs(change*mass_stoic.hydrogen) <= st.hydrogen || error("Hydrogen is not
↪  sufficient for this goal")
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                butanol = st.butanol + change*mass_stoic.butanol,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                co2 = st.co2 + change*mass_stoic.co2,
                water = st.water + change*mass_stoic.water))
end

function pyruv_to_acetone(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -2, water = -1, co2 = +3, hydrogen = +2, acetone = +1)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                water = stoic.water*m_water(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                co2 = stoic.co2*m_co2(),
                acetone = stoic.acetone*m_acetone())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪  goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                water = st.water + change*mass_stoic.water,
                co2 = st.co2 + change*mass_stoic.co2,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                acetone = st.acetone + change*mass_stoic.acetone))
end

function pyruv_to_lact(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -1, hydrogen = -1, lactate = +1)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                lactate = stoic.lactate*m_lactate())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪  goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
abs(change*mass_stoic.hydrogen) <= st.hydrogen || error("Hydrogen is not
↪  sufficient for this goal")
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                lactate = st.lactate + change*mass_stoic.lactate))
end


function pyruv_to_succin(st; goal = (; pyruvate = 0.0))
stoic = (pyruvate = -1, co2 = -1, hydrogen = -2, succinate = +1, water =
↪  +1)
mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                co2 = stoic.co2*m_co2(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                succinate = stoic.succinate*m_succinate(),
                water = stoic.water*m_water())
goal.pyruvate <= st.pyruvate || error("Pyruvate is not sufficient for this
↪   goal")
change = (goal.pyruvate - st.pyruvate)/mass_stoic.pyruvate
abs(change*mass_stoic.hydrogen) <= st.hydrogen || error("Hydrogen is not
↪   sufficient for this goal")
abs(change*mass_stoic.co2) <= st.co2 || error("CO2 is not sufficient for
↪   this goal")
new_st = merge(st,
                (pyruvate = goal.pyruvate,
                co2 = st.co2 + change*mass_stoic.co2,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                succinate = st.succinate + change*mass_stoic.succinate,
                water = st.water + change*mass_stoic.water))
end
```

5 Other pathways stemming from glycolysis

However, there are also some other important reactions that are in these pathways as the above products are in some cases intermediates for the production of something else. The reactions taking some of these products and converting them to other products are:

- Acetaldehyde + H2 -> Ethanol
- Lactate + H2 -> Propionate
- Succinate + CO2 -> Propionate
- Formate <-> CO2 + H2
- Propionate + 2CO2 + 6H2 -> Valerate

and the code for their implementation can be seen below

```
function acetaldehyde_to_ethanol(st; goal = (; acetaldehyde = 0.0))
stoic = (acetaldehyde = -1, hydrogen = -1, ethanol = +1)
mass_stoic = (acetaldehyde = stoic.acetaldehyde*m_acetaldehyde(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                ethanol = stoic.ethanol*m_ethanol())
goal.acetaldehyde <= st.acetaldehyde || error("Acetaldehyde is not
↪   sufficient for this goal")
change = (goal.acetaldehyde - st.acetaldehyde)/mass_stoic.acetaldehyde
abs(change*mass_stoic.hydrogen) <= st.hydrogen || error("Hydrogen is not
↪   sufficient for this goal")
new_st = merge(st,
                (acetaldehyde = goal.acetaldehyde,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                ethanol = st.ethanol + change*mass_stoic.ethanol))
end


function lact_to_propionate(st; goal = (; lactate = 0.0))
stoic = (lactate = -1, hydrogen = -1, propionate = +1)
mass_stoic = (lactate = stoic.lactate*m_lactate(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                propionate = stoic.propionate*m_propionate())
goal.lactate <= st.lactate || error("Lactate is not sufficient for this
↪  goal")
change = (goal.lactate - st.lactate)/mass_stoic.lactate
abs(change*mass_stoic.hydrogen) <= st.hydrogen || error("Hydrogen is not
↪  sufficient for this goal")
new_st = merge(st,
                (lactate = goal.lactate,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                propionate = st.propionate + change*mass_stoic.propionate))
end


function succin_to_propionate(st; goal = (; succinate = 0.0))
stoic = (succinate = -1, propionate = +1, co2 = +1)
mass_stoic = (succinate = stoic.succinate*m_succinate(),
                propionate = stoic.propionate*m_propionate(),
                co2 = stoic.co2*m_co2())
goal.succinate <= st.succinate || error("Succinate is not sufficient for
↪  this goal")
change = (goal.succinate - st.succinate)/mass_stoic.succinate
new_st = merge(st,
                (succinate = goal.succinate,
                propionate = st.propionate + change*mass_stoic.propionate,
                co2 = st.co2 + change*mass_stoic.co2))
end

# The formate balance isn't exactly like all the other reactions where
# the goal is the main reactant. It is a reaction very close to
# equilibrium that in pH near neutral or higher is favored on
# formate. If you expect that formate will be produced, you can give a
# goal that formate has this concentration and it will remove enough
# co2 and hydrogen for it to be feasible. Since it is common for none
# to be produced, the default value will be expect that none will be
# produced.
function formate_balance(st; goal = (; formate = 0.0))
stoic = (co2 = -1, hydrogen = -1, formate = +1)
mass_stoic = (co2 = stoic.co2*m_co2(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                formate = stoic.formate*m_formate())
change = (goal.formate - st.formate)/mass_stoic.formate
abs(change*mass_stoic.hydrogen) <= st.hydrogen || error("Hydrogen is not
↪  sufficient for this goal")
abs(change*mass_stoic.co2) <= st.co2 || error("CO2 is not sufficient for
↪  this goal")
new_st = merge(st,
                (formate = goal.formate,
                co2 = st.co2 + change*mass_stoic.co2,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end


function propionate_to_valerate(st; goal = (; valerate = 0.0))
stoic = (propionate = -1, co2 = -2, hydrogen = -6, valerate =+1)
mass_stoic = (propionate = stoic.propionate*m_propionate(),
                co2 = stoic.co2*m_co2(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                valerate = stoic.valerate*m_valerate())
goal.valerate <= m_valerate()*st.propionate/m_propionate() ||
↪  error("Propionate is not sufficient for this goal")
change = (goal.valerate - st.valerate)/mass_stoic.valerate
new_st = merge(st,
                (valerate = goal.valerate,
                propionate = st.propionate + change*mass_stoic.propionate,
                co2 = st.co2 + change*mass_stoic.co2,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end
```

With these implemented, we might want to write down the complete reaction of pyruvate to ethanol since we know it can be done through acetaldehyde. This is a rather simple implementation as it just sequentially runs the two functions.

```
function pyruv_to_ethanol(st; pyr_goal = (; pyruvate = 0.0),
                        acet_goal = (; acetaldehyde = 0.0))
acetaldehyde_st = pyruv_to_acetaldehyde(st, goal = pyr_goal)
new_st = acetaldehyde_to_ethanol(acetaldehyde_st, goal = acet_goal)
end
```

A more complex one is the pathway that goes from pyruvate to propionate. Propionate can be produced from lactate as the intermediate or from succinate, with both having the same end result. We can write a more complex composition function which takes both pathways and the extent to which each is followed, which might be of interest. For this implementation, we follow a similar logic as above, with one more important step. One of our inputs is the amount of pyruvate that goes to lactate production (since there are two pathways, the other is 1-lactate). Since we know how much pyruvate goes to each reaction, we can change the goal of each function to not consume all the pyruvate, but only the one we define. If we want all the pyruvate to be consumed by this reaction and we want each intermediate to only take an amount, this is simple as it is just the initial pyruvate times the amount. However, since this compound reaction might be used in other larger composition reactions, we want a behaviour that works even if the pyruvate goal of the total reaction is non zero. This expression turns out to be [st.pyruvate - (st.pyryvate - goal.pyruvate)*lact$_{\text{amount}}$] and is used extensively below in all compound pathways that will be defined. Another important thing in this function is the final composition. Since reactions don't occur serially but simultaneously, we need to merge them together. However, in the case where what is being created in the reaction already existed in the reactor, each state will have the initial amount and add to it what was produced/consumed in it. Therefore, to get correct results, if 2 (or more in other more complex pathways) reactions produce the same thing, we must always substract the initial value to not inflate them. The definition can be seen below.

```
function pyruv_to_propionate(st, lact_amount; pyr_goal = (; pyruvate =
↪  0.0),
                            succin_goal = (; succinate = 0.0),
                            lact_goal = (; lactate = 0.0))
lact_prod_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  pyr_goal.pyruvate)*lact_amount))
succin_prod_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  pyr_goal.pyruvate)*(1-lact_amount)))

lact_st = pyruv_to_lact(st, goal = lact_prod_goal)
succin_st = pyruv_to_succin(st, goal = succin_prod_goal)
new_st = merge(st,
                (pyruvate = pyr_goal.pyruvate,
                    hydrogen = st.hydrogen - (st.hydrogen -
                    ↪  succin_st.hydrogen) - (st.hydrogen -
                    ↪  lact_st.hydrogen),
                    co2 = succin_st.co2,
                    succinate = succin_st.succinate,
                    lactate = lact_st.lactate))

prop_st1 = lact_to_propionate(new_st, goal = lact_goal)
prop_st2 = succin_to_propionate(new_st, goal = succin_goal)

final_st = merge(new_st,
                    (lactate = lact_goal.lactate,
                    succinate = succin_goal.succinate,
                    propionate = prop_st1.propionate + prop_st2.propionate
                    ↪  - new_st.propionate,
                    hydrogen = prop_st1.hydrogen,
                    co2 = prop_st2.co2))
end
```

6 Other pathways for glucose consumption

However, the glycolytic pathway for pyruvate production and its conversion to products isn't the only possible route. There are also other pathways for the consumption of glucose.

6.1 Heterolactic fermentation

One such pathway is the PK pathway where glucose is converted to one mole of glyceraldehyde-3-phosphate (which is then converted to pyruvate) and one mole of acetyl-CoA. This route produces 3 hydrogen moles together with those, which means that reductions are heavily favored. For this reason, the pyruvate produced is converted to lactate and acetyl-CoA favors the reductive pathway of ethanol production instead of acetate, although acetate can be seen in this pathway. This is also called the heterolactic fermentation pathway due to how lactate is produced together with a co-product. The two primitive reactions for heterolactate with ethanol and acetate are defined and then a compound reaction that combines them.

```
function ethanol_heterolactate(st; goal = (; glucose = 0.0))
stoic = (glucose = -1, pyruvate = +1, ethanol = +1, hydrogen = +1, co2 =
↪  +2)
mass_stoic = (glucose = stoic.glucose*m_glucose(),
                pyruvate = stoic.pyruvate*m_pyruvate(),
                ethanol = stoic.ethanol*m_ethanol(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                co2 = stoic.co2*m_co2())
goal.glucose <= st.glucose || error("Glucose is not sufficient for this
↪  goal")
change = (goal.glucose - st.glucose)/mass_stoic.glucose
pyr_st = merge(st,
                (glucose = goal.glucose,
                pyruvate = st.pyruvate + change*mass_stoic.pyruvate,
                ethanol = st.ethanol + change*mass_stoic.ethanol,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                co2 = st.co2 + change*mass_stoic.co2))

new_st = pyruv_to_lact(pyr_st)
end

function acetate_heterolactate(st; goal = (; glucose = 0.0))
stoic = (glucose = -1, pyruvate = +1, acetate = +1, hydrogen = +3, co2 =
↪  +2)
mass_stoic = (glucose = stoic.glucose*m_glucose(),
                pyruvate = stoic.pyruvate*m_pyruvate(),
                acetate = stoic.acetate*m_acetate(),
                hydrogen = stoic.hydrogen*m_hydrogen(),
                co2 = stoic.co2*m_co2())
goal.glucose <= st.glucose || error("Glucose is not sufficient for this
↪  goal")
change = (goal.glucose - st.glucose)/mass_stoic.glucose
```

```
pyr_st = merge(st,
                (glucose = goal.glucose,
                pyruvate = st.pyruvate + change*mass_stoic.pyruvate,
                acetate = st.acetate + change*mass_stoic.acetate,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen,
                co2 = st.co2 + change*mass_stoic.co2))

new_st = pyruv_to_lact(pyr_st)
end


function heterolactic_ferment(st; goal = (; glucose = 0.0),
                            acet_amount = 0)
acet_prod_goal = (; glucose = (st.glucose - (st.glucose -
↪ goal.glucose)*acet_amount))
eth_prod_goal = (; glucose = (st.glucose - (st.glucose -
↪ goal.glucose)*(1-acet_amount)))

eth_st = ethanol_heterolactate(st, goal)
acet_st = acetate_heterolactate(st, goal)
new_st = merge(st,
                (glucose = goal.glucose,
                ethanol = eth_st.ethanol,
                acetate = acet_st.acetate,
                lactate = eth_st.lactate + acet_st.lactate - st.lactate,
                co2 = eth_st.co2 + acet_st.co2 - st.co2,
                hydrogen = eth_sth.hydrogen + acet_st.hydrogen -
                ↪ st.hydrogen))
end
```

6.2 Bifidus fermentation

Another possible pathway is bifidus fermentation where 1/4th of the glucose is converted immediately to acetyl-CoA (half a molecule) and the rest of the carbons (5) go through a different pathway to pyruvate and acetyl-CoA. However, in this process, only one hydrogen is produced (oxidation of glyceraldehyde-3-phosphate to pyruvate) so pathways of reduction are not as favored and acetyl-CoA is converted to acetate. The common co-product however remains lactate using the one hydrogen created from pyruvate.

```
function bifidus_ferment(st; goal = (; glucose = 0.0))
stoic = (glucose = -1, acetate = +1.5, pyruvate = +1, hydrogen = +1)
mass_stoic = (glucose = stoic.glucose*m_glucose(),
                acetate = stoic.acetate*m_acetate(),
                pyruvate = stoic.pyruvate*m_pyruvate(),
                hydrogen = stoic.hydrogen*m_hydrogen())
goal.glucose <= st.glucose || error("Glucose is not sufficient for this
↪ goal")
change = (goal.glucose - st.glucose)/mass_stoic.glucose
```

```
pyr_st = merge(st,
               (glucose = goal.glucose,
               pyruvate = st.pyruvate + change*mass_stoic.pyruvate,
               acetate = st.acetate + change*mass_stoic.acetate,
               hydrogen = st.hydrogen + change*mass_stoic.hydrogen))

new_st = pyruv_to_lact(pyr_st)
end
```

6.3 Ethanol fermentation

Another common pathway of glucose consumption is the ethanol fermentation which happens in yeasts. This is not very common in a typical mixed culture, but is added here for completion purposes and due to how easy it is to implement.

```
function ethanol_fermentation(st; goal = (; glucose = 0.0))
stoic = (glucose = -1, ethanol = +2, co2 = +2)
mass_stoic = (glucose = stoic.glucose*m_glucose(),
               ethanol = stoic.ethanol*m_ethanol(),
               co2 = stoic.co2*m_co2())
goal.glucose <= st.glucose || error("Glucose is not sufficient for this
↪   goal")
change = (goal.glucose - st.glucose)/mass_stoic.glucose
new_st = merge(st,
               (glucose = goal.glucose,
               ethanol = st.ethanol + change*mass_stoic.ethanol,
               co2 = st.co2 + change*mass_stoic.co2))
end
```

6.4 Glucose consumption

Having defined 4 different pathways in which glucose is consumed, there is interest in defining a glucose consumption function which given the amount of glucose in each pathway can calculate the products. Then, this can be linked to pyruvate consumption pathways to final products. This is shown here.

```
function glucose_consumption(st, bifidus_amount, eth_amount,
                             heterolact_amount; goal = (; glucose = 0.0),
                             acet_amount = 0)
glycolysis_amount = 1-bifidus_amount-eth_amount-heterolact_amount
bifidus_goal = (; glucose = (st.glucose - (st.glucose -
↪   goal.glucose))*bifidus_amount)
eth_goal = (; glucose = (st.glucose - (st.glucose -
↪   goal.glucose))*eth_amount)
heterolact_goal = (; glucose = (st.glucose - (st.glucose -
↪   goal.glucose))*heterolact_amount)
```

```
glycolysis_goal = (; glucose = (st.glucose - (st.glucose -
↪  goal.glucose))*glycolysis_amount)

bifidus_st = bifidus_ferment(st, goal)
eth_st = ethanol_fermentation(st, goal)
heterolact_st = heterolactic_ferment(st, goal = goal, acet_amount =
↪  acet_amount)
glycolysis_st = glycolysis(st, goal)

new_st = merge(st,
                (glucose = goal.glucose,
                pyruvate = glycolysis_st.pyruvate,
                acetate = heterolact_st.acetate + bifidus_st.acetate -
                ↪  st.acetate,
                ethanol = heterolact_st.ethanol + eth_st.ethanol -
                ↪  st.ethanol,
                lactate = heterolact_st.lactate + bifidus_st.lactate -
                ↪  st.lactate,
                co2 = heterolact_st.co2 + eth_st.co2 - st.co2,
                hydrogen = glycolysis_st.hydrogen +
                ↪  heterolact_st.hydrogen))
end
```

6.5 Aerobic consumption

In aerobic conditions (with oxygen in the reactor), glucose goes down the glycolytic pathway, but pyruvate is not converted into any of the aforementioned products, but rather enters the Krebs cycle where it continuously produces energy, CO_2 and hydrogen. Pyruvate is first converted to acetyl-CoA in this process (which abstracting the details of CoA can be simulated with the conversion to acetate we have written down) and then that breaks down to CO_2 and hydrogen in the Krebs cycle. NADˆ+ and FAD is required for this process and for it to be in its oxidized state all the time, this process needs to be done with oxygen. By the electron balance of these, 2 moles of oxygen are required per run of the Krebs cycle. Also 2 moles of water are necessary. Therefore, in our reaction system which abstracts the details of each pathway, it could be described as $CH3COOH + 2H_2O \xrightarrow{2O_2} 2CO_2 + 4H_2$ where the Oxygen is written in the arrow becuase the redox reactions it participates in are hidden. Kreb's cycle operates for as long as there is oxygen in the reactor, therefore, the goal of this function will be oxygen to 0. However, if written as two reactions (pyruvate to acetate and acetate oxidation), we will get incorrect results, as we want pyruvate to acetate for as much as oxygen can be produced. Therefore, we need to combine the reactions. The implementation is shown below

```
function aerobic_pyruvate_oxidation(st; goal = (; oxygen = 0.0))
    stoic = (pyruvate = -1, water = -3, oxygen = -2, co2 = +3, hydrogen =
    ↪  +5)
    mass_stoic = (pyruvate = stoic.pyruvate*m_pyruvate(),
                  water = stoic.water*m_water(),
                  oxygen = stoic.oxygen*m_oxygen(),
                  co2 = stoic.co2*m_co2(),
                  hydrogen = stoic.hydrogen*m_hydrogen())
    goal.oxygen <= st.oxygen || error("Oxygen is not sufficient for this
    ↪  goal")
    change = (goal.oxygen - st.oxygen)/mass_stoic.oxygen
    abs(change*mass_stoic.pyruvate) <= st.pyruvate || error("Pyruvate is
    ↪  not sufficient for this goal")
    new_st = merge(st,
                   (oxygen = goal.oxygen,
                    pyruvate = st.pyruvate + change*mass_stoic.pyruvate,
                    water = st.water + change*mass_stoic.water,
                    co2 = st.co2 + change*mass_stoic.co2,
                    hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end
```

7 Acetogenic routes

Another big part of a typical anaerobic mixed culture fermentation is acetogenesis, the process in which various materials (such as propionate, butyrate, lactate, ethanol etc.) are converted to acetate. These processes are described below.

```
function propionate_to_acetate(st; goal = (; propionate = 0.0))
stoic = (propionate = -1, water = -2, acetate = +1, co2 = +1, hydrogen =
↪  +3)
mass_stoic = (propionate = stoic.propionate*m_propionate(),
                water = stoic.water*m_water(),
                acetate = stoic.acetate*m_acetate(),
                co2 = stoic.co2*m_co2(),
                hydrogen = stoic.hydrogen*m_hydrogen())
goal.propionate <= st.propionate || error("Propionate is not sufficient
↪  for this goal")
change = (goal.propionate - st.propionate)/mass_stoic.propionate
new_st = merge(st,
                (propionate = goal.propionate,
                water = st.water + change*mass_stoic.water,
                acetate = st.acetate + change*mass_stoic.acetate,
                co2 = st.co2 + change*mass_stoic.co2,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end

function butyr_to_acetate(st; goal = (; butyrate = 0.0))
stoic = (butyrate = -1, water = -2, acetate = +2, hydrogen = +2)
mass_stoic = (butyrate = stoic.butyrate*m_butyrate(),
                water = stoic.water*m_water(),
                acetate = stoic.acetate*m_acetate(),
                hydrogen = stoic.hydrogen*m_hydrogen())
goal.butyrate <= st.butyrate || error("Butyrate is not sufficient for this
↪  goal")
```

```
change = (goal.butyrate - st.butyrate)/mass_stoic.butyrate
new_st = merge(st,
                (butyrate = goal.butyrate,
                water = st.water + change*mass_stoic.water,
                acetate = st.acetate + change*mass_stoic.acetate,
                hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end

function ethanol_to_acetate(st; goal = (; ethanol = 0.0))
    stoic = (ethanol = -1, water = -2, acetate = +1, hydrogen = +2)
    mass_stoic = (ethanol = stoic.ethanol*m_ethanol(),
                  water = stoic.water*m_water(),
                  acetate = stoic.acetate*m_acetate(),
                  hydrogen = stoic.hydrogen*m_hydrogen())
    goal.ethanol <= st.ethanol || error("Ethanol is not sufficient for
    ↪  this goal")
    change = (goal.ethanol - st.ethanol)/mass_stoic.ethanol
    new_st = merge(st,
                   (ethanol = goal.ethanol,
                    water = st.water + change*mass_stoic.water,
                    acetate = st.acetate + change*mass_stoic.acetate,
                    hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end

function lact_to_acetate(st; goal = (; lactate = 0.0))
    stoic = (lactate = -1, water = -1, acetate = +1, hydrogen = +2, co2 =
    ↪  +1)
    mass_stoic = (lactate = stoic.lactate*m_lactate(),
                  water = stoic.water*m_water(),
                  acetate = stoic.acetate*m_acetate(),
                  co2 = stoic.co2*m_co2(),
                  hydrogen = stoic.hydrogen*m_hydrogen())
    goal.lactate <= st.lactate || error("Lactate is not sufficient for
    ↪  this goal")
    change = (goal.lactate - st.lactate)/mass_stoic.lactate
    new_st = merge(st,
                   (lactate = goal.lactate,
                    water = st.water + change*mass_stoic.water,
                    acetate = st.acetate + change*mass_stoic.acetate,
                    co2 = st.co2 + change*mass_stoic.co2,
                    hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end

# In some cases, lactate acetogenesis can also happen together with
# its reduction to propionate.

function lact_to_acet_prop(st; goal = (; lactate = 0.0))
    stoic = (lactate = -2, acetate = +1, propionate = +1, hydrogen = +1,
    ↪  co2 = +1)
    mass_stoic = (lactate = stoic.lactate*m_lactate(),
                  acetate = stoic.acetate*m_acetate(),
                  co2 = stoic.co2*m_co2(),
                  propionate = stoic.propionate*m_propionate(),
                  hydrogen = stoic.hydrogen*m_hydrogen())
    goal.lactate <= st.lactate || error("Lactate is not sufficient for
    ↪  this goal")
    change = (goal.lactate - st.lactate)/mass_stoic.lactate
    new_st = merge(st,
                   (lactate = goal.lactate,
                    acetate = st.acetate + change*mass_stoic.acetate,
                    propionate = st.propionate +
                    ↪  change*mass_stoic.propionate,
                    hydrogen = st.hydrogen + change*mass_stoic.hydrogen))
end

# For this reason, we can also define a compound reaction that lists
# both pathways of lactate acetogenesis (conversion to acetate or
# conversion to a mixture of it and propionate due to the surplus of
# hydrogen allowing for lactate reduction) with the extent to which
# the propionate producing reaction happens.

function lactate_acetogenesis(st, prop_amount; goal = (; lactate = 0.0))
    acet_prod_goal = (; lactate = (st.lactate - (st.lactate -
    ↪  goal.lactate)*(1-prop_amount)))
    prop_prod_goal = (; lactate = (st.lactate - (st.lactate -
    ↪  goal.lactate)*prop_amount))

    acet_st = lact_to_acetate(st, goal = acet_prod_goal)
    prop_st = lact_to_acet_prop(st, goal = prop_prod_goal)
    new_st = merge(st,
                   (lactate = goal.lactate,
                    acetate = acet_st.acetate + prop_st.acetate -
                    ↪  st.acetate,
                    hydrogen = acet_st.hydrogen + prop_st.hydrogen -
                    ↪  st.hydrogen,
                    co2 = acet_st.co2 + prop_st.co2 - st.co2,
                    propionate = prop_st.propionate,
                    water = acet_st.water))
end

function homoacetogenic_acetate(st; goal = (; hydrogen = 0.0))
    stoic = (hydrogen = -4, co2 = -2, acetate = +1, water = +1)
    mass_stoic = (hydrogen = stoic.hydrogen*m_hydrogen(),
                  co2 = stoic.co2*m_co2(),
                  acetate = stoic.acetate*m_acetate(),
                  water = stoic.water*m_water())
    goal.hydrogen <= st.hydrogen || error("Hydrogen is not sufficient for
    ↪  this goal")
    abs(change*mass_stoic.co2) <= st.co2 || error("CO2 is not sufficient
    ↪  for this goal")
    change = (goal.hydrogen - st.hydrogen)/mass_stoic.hydrogen
    new_st = merge(st,
                   (hydrogen = goal.hydrogen,
                    co2 = st.co2 + change*mass_stoic.co2,
                    acetate = st.acetate + change*mass_stoic.acetate,
                    water = st.water + change*mass_stoic.water))
end
```

After this is done, we can define a large compound reaction for acetogenesis that given the final mass of each material can find the final state that is reached.

```
function acetogenesis(st; prop_goal = (; propionate = st.propionate),
                      butyr_goal = (; butyrate = st.butyrate),
                      eth_goal = (; ethanol = st.ethanol),
                      lact_goal = (; lactate = st.lactate),
                      hyd_goal = (; hydrogen = st.hydrogen),
                      lact_prop = 0)
    prop_st = propionate_to_acetate(st, goal = prop_goal)
    butyr_st = butyr_to_acetate(st, goal = butyr_goal)
    eth_st = ethanol_to_acetate(st, goal = eth_goal)
    lact_st = lactate_acetogenesis(st, lact_prop, goal = lact_goal)

    new_st = merge(st,
                   (propionate = prop_st.propionate + lact_st.propionate -
                   ↪  st.propionate,
                    butyrate = butyr_st.butyrate,
                    ethanol = eth_st.ethanol,
                    lactate = lact_st.lactate,
                    co2 = prop_st.co2 + lact_st.co2 - st.co2,
                    water = prop_st.water + butyr_st.water + eth_st.water
                    ↪  + lact_st.water - 3st.water,
                    acetate = prop_st.acetate + butyr_st.acetate +
                    ↪  eth_st.acetate + lact_st.acetate - 3st.acetate,
                    hydrogen = prop_st.hydrogen + butyr_st.hydrogen +
                    ↪  eth_st.hydrogen + lact_st.hydrogen - 3st.hydrogen))

    #homoacetic_st = homoacetogenic_acetate(new_st, goal = hyd_goal)
end
```

8 Common fermentative pathways

After defining all the above, there is a lot of interest in some very common compound fermentative pathways. For example, we know from literature that a very common pathway is that ethanol and acetate are produced in equimolar amounts, so instead of writing that in the final function we want to use, we can implement it directly and then use this in the final function that describes the combination of pathways we assume to be followed. Besides acetate-ethanol fermentation, we know that a fermentation of butyrate with acetate in a 3:1 molar analogy is common and propionate-acetate in a 2:1.

```
function acetate_ethanol_fermentation(st; goal = (; pyruvate = 0.0))
acet_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate)*0.5))
eth_goal = (; pyruvate = (st.pyruvate - (st.pyruvate - goal.pyruvate)*0.5))

acet_st = pyruv_to_acetate(st, goal = acet_goal)
eth_st = pyruv_to_ethanol(st, pyr_goal = eth_goal)

new_st = merge(st,
                (pyruvate = goal.pyruvate,
                acetate = acet_st.acetate,
                ethanol = eth_st.ethanol,
                co2 = acet_st.co2 + eth_st.co2 - st.co2,
                water = acet_st.water))
end

function acetate_butyrate_fermentation(st; goal = (; pyruvate = 0.0))
acet_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate)*0.25))
butyr_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate)*0.75))

acet_st = pyruv_to_acetate(st, goal = acet_goal)
butyr_st = pyruv_to_butyr(st, goal = butyr_goal)

new_st = merge(st,
                (pyruvate = goal.pyruvate,
                acetate = acet_st.acetate,
                butyrate = butyr_st.butyrate,
                hydrogen = acet_st.hydrogen + butyr_st.hydrogen -
                ↪  st.hydrogen,
                co2 = acet_st.co2 + butyr_st.co2 - st.co2))
end

# Reminder that the pyruvate to propionate function has levers for how
# much lactate was produced from each pathway and if lactate or
# succinate are accumulated in the reactor. In the case of
# acetate-propionate fermentation with this stoichiometry, propionate
# is fully converted so these aren't necessary. More complex ones can
# be defined to explain accumulation of lactate and succinate, but
# this is the standard acidogenic route.
function acetate_propionate_fermentation(st; pyr_goal = (; pyruvate =
↪  0.0), lact_goal=(; lactate = 0.0), lact_amount = 1, prop_amount = 2/3)
    acet_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
    ↪  pyr_goal.pyruvate)*(1-prop_amount)))
    prop_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
    ↪  pyr_goal.pyruvate)*prop_amount))

    acet_st = pyruv_to_acetate(st, goal = acet_goal)
    prop_st = pyruv_to_propionate(st, lact_amount, pyr_goal = prop_goal,
    ↪  lact_goal = lact_goal)

    new_st = merge(st,
                   (pyruvate = pyr_goal.pyruvate,
                    acetate = acet_st.acetate,
                    propionate = prop_st.propionate,
                    co2 = acet_st.co2,
                    lactate = prop_st.lactate,
                    hydrogen = acet_st.hydrogen + prop_st.hydrogen -
                    ↪  st.hydrogen))
end
```

Besides these, there is another one, which I have not seen in literature but I have needed to accurately describe my experiments, which is the ethanol-propionate pathway. Due to both ethanol and propionate being pathways that need multiple reductions, this ends up needing a lot of hydrogen and as such is not commonly reported. But if there is a surplus of hydrogen from other processes, it can be used to describe a system and therefore is implemented here.

```
function ethanol_propionate_fermentation(st; pyr_goal = (; pyruvate = 0.0),
                                         lact_goal = (; lactate = 0.0),
                                         succin_goal = (; succinate = 0.0),
                                         prop_amount = 0.5, lact_amount =
                                         ↪  1)
    eth_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
    ↪  pyr_goal.pyruvate)*(1-prop_amount)))
    prop_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
    ↪  pyr_goal.pyruvate)*prop_amount))

    eth_st = pyruv_to_ethanol(st, pyr_goal = eth_goal)
    prop_st = pyruv_to_propionate(st, lact_amount, pyr_goal = prop_goal,
    ↪  lact_goal = lact_goal, succin_goal = succin_goal)

    new_st = merge(st,
                   (pyruvate = pyr_goal.pyruvate,
                    ethanol = eth_st.ethanol,
                    propionate = prop_st.propionate,
                    lactate = prop_st.lactate,
                    co2 = eth_st.co2,
                    hydrogen = eth_st.hydrogen + prop_st.hydrogen -
                    ↪  st.hydrogen,
                    succinate = prop_st.succinate))
end
```

8.1 ABE Fermentation

Another common pathway studied in literature is ABE fermentation. In this, there is an acidogenic phase where mostly acetate and butyrate are produced (with some ethanol production) and after some time, the culture enters the solventogenic phase where ethanol is produced with a higher yield and together with acetone and butanol. This system is described with the below, fairly complex, function. Note that the extent to which each product is produced is not set in stone for ABE fermentation so all of these are given as keyword arguments.

```
function ABE_fermentation(st; goal = (; pyruvate = 0.0),
                        acet_amount, aceteth_amount, butyr_amount,
                        solveth_amount, acetone_amount)
butanol_amount = 1 - acet_amount - aceteth_amount - butyr_amount -
↪  solveth_amount - acetone_amount
acet_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate))*acet_amount)
aceteth_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate))*aceteth_amount)
butyr_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate))*butyr_amount)
solveth_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate))*solveth_amount)
acetone_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate))*acetone_amount)
butanol_goal = (; pyruvate = (st.pyruvate - (st.pyruvate -
↪  goal.pyruvate))*butanol_amount)

acet_st = pyruv_to_acetate(st, goal = acet_goal)
aceteth_st = acetate_ethanol_fermentation(st, goal = aceteth_goal)
butyr_st = pyruv_to_butyr(st, goal = butyr_goal)
acidogenic_st = merge(st,
                        (pyruvate = acet_st.pyruvate + aceteth_st.pyruvate
                        ↪  + butyr_st.pyruvate - 2st.pyruvate,
                        acetate = acet_st.acetate + aceteth_st.acetate -
                        ↪  st.acetate,
                        butyrate = butyr_st.butyrate,
                        ethanol = aceteth_st.ethanol,
                        hydrogen = acet_st.hydrogen,
                        co2 = acet_st.co2 + aceteth_st.co2 + butyr_st.co2
                        ↪  - 2st.co2,
                        water = acet_st.water + aceteth_st.water -
                        ↪  st.water))

solveth_st = pyruv_to_ethanol(acidogenic_st, goal = solveth_goal)
acetone_st = pyruv_to_acetone(acidogenic_st, goal = acetone_goal)
butanol_st = pyruv_to_butanol(acidogenic_st, goal = butanol_goal)
```

```
solventogenic_st = merge(acidogenic_st,
                         (pyruvate = goal.pyruvate,

                         ethanol = solveth_st.ethanol,
                         acetone = acetone_st.acetone,
                         butanol = butanol_st.butanol,
                         co2 = solveth_st.co2 + acetone_st.co2 +
                         ↪  butanol_st.co2 - 2acidogenic_st.co2,
                         hydrogen = acetone_st.hydrogen +
                         ↪  butanol_st.hydrogen -
                         ↪  acidogenic_st.hydrogen,
                         water = acetone_st.water + butanol_st.water -
                         ↪  acidogenic_st.water))
end
```

9 Test on experiments

This section is about a test for the experiments we have done to see if we can validate the data. Obviously, we first need to import dependencies and our data.

```
using DrWatson
@quickactivate "Masters_Thesis"
include(srcdir("filenames.jl"))
include(srcdir("metabolic_pathways", "primitives.jl"))
include(srcdir("metabolic_pathways", "core_pathways.jl"))
include(srcdir("metabolic_pathways", "compound_pathways.jl"))
include(srcdir("metabolic_pathways", "acetogenesis.jl"))
using CSV, DataFrames

# Read all the data
exp_35 = "10_11"
exp_40 = "28_11"
mix_amount = ["0", "1", "2", "4", "8"]

# Experiment @35 C
df35_0 = CSV.read(get_conc_csv(exp_35, mix_amount[1]), DataFrame)
df35_1 = CSV.read(get_conc_csv(exp_35, mix_amount[2]), DataFrame)
df35_2 = CSV.read(get_conc_csv(exp_35, mix_amount[3]), DataFrame)
df35_4 = CSV.read(get_conc_csv(exp_35, mix_amount[4]), DataFrame)
df35_8 = CSV.read(get_conc_csv(exp_35, mix_amount[5]), DataFrame)

# Experiment @40 C
df40_0 = CSV.read(get_conc_csv(exp_40, mix_amount[1]), DataFrame)
df40_1 = CSV.read(get_conc_csv(exp_40, mix_amount[2]), DataFrame)
df40_2 = CSV.read(get_conc_csv(exp_40, mix_amount[3]), DataFrame)
df40_4 = CSV.read(get_conc_csv(exp_40, mix_amount[4]), DataFrame)
df40_8 = CSV.read(get_conc_csv(exp_40, mix_amount[5]), DataFrame)
```

Then, we need to initialize the named tuples this framework uses instead of the data frames.

```
# 35 C
v = 0.8
init_st35_0 = (sucrose = df35_0.Sucrose[1], glucose = df35_0.Glucose[1],
            fructose = df35_0.Fructose[1], lactate = df35_0.Lactate[2],
            acetate = df35_0.Acetate[1], propionate = df35_0.Propionate[2],
            ethanol = df35_0.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass35_0 = conc_to_mass(init_st35_0, v)

init_st35_1 = (sucrose = df35_1.Sucrose[1], glucose = df35_1.Glucose[1],
            fructose = df35_1.Fructose[1], lactate = df35_1.Lactate[2],
            acetate = df35_1.Acetate[2], propionate = df35_1.Propionate[2],
            ethanol = df35_1.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass35_1 = conc_to_mass(init_st35_1, v)

init_st35_2 = (sucrose = df35_2.Sucrose[1], glucose = df35_2.Glucose[1],
            fructose = df35_2.Fructose[1], lactate = df35_2.Lactate[1],
            acetate = df35_2.Acetate[1], propionate = df35_2.Propionate[1],
            ethanol = df35_2.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass35_2 = conc_to_mass(init_st35_2, v)

init_st35_4 = (sucrose = df35_4.Sucrose[1], glucose = df35_4.Glucose[1],
            fructose = df35_4.Fructose[1], lactate = df35_4.Lactate[1],
            acetate = df35_4.Acetate[2], propionate = df35_4.Propionate[1],
            ethanol = df35_4.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass35_4 = conc_to_mass(init_st35_4, v)

init_st35_8 = (sucrose = df35_8.Sucrose[1], glucose = df35_8.Glucose[1],
            fructose = df35_8.Fructose[1], lactate = df35_8.Lactate[1],
            acetate = df35_8.Acetate[2], propionate = df35_8.Propionate[1],
            ethanol = df35_8.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass35_8 = conc_to_mass(init_st35_8, v)

# 40C
v = 0.8
init_st40_0 = (sucrose = df40_0.Sucrose[1], glucose = df40_0.Glucose[1],
            fructose = df40_0.Fructose[1], lactate = df40_0.Lactate[2],
            acetate = df40_0.Acetate[1], propionate = df40_0.Propionate[2],
            ethanol = df40_0.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass40_0 = conc_to_mass(init_st40_0, v)

init_st40_1 = (sucrose = df40_1.Sucrose[1], glucose = df40_1.Glucose[1],
            fructose = df40_1.Fructose[1], lactate = df40_1.Lactate[2],
            acetate = df40_1.Acetate[2], propionate = df40_1.Propionate[2],
            ethanol = df40_1.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass40_1 = conc_to_mass(init_st40_1, v)

init_st40_2 = (sucrose = df40_2.Sucrose[1], glucose = df40_2.Glucose[1],
            fructose = df40_2.Fructose[1], lactate = df40_2.Lactate[1],
            acetate = df40_2.Acetate[1], propionate = df40_2.Propionate[1],
            ethanol = df40_2.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass40_2 = conc_to_mass(init_st40_2, v)

init_st40_4 = (sucrose = df40_4.Sucrose[1], glucose = df40_4.Glucose[1],
            fructose = df40_4.Fructose[1], lactate = df40_4.Lactate[1],
            acetate = df40_4.Acetate[2], propionate = df40_4.Propionate[1],
            ethanol = df40_4.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass40_4 = conc_to_mass(init_st40_4, v)

init_st40_8 = (sucrose = df40_8.Sucrose[1], glucose = df40_8.Glucose[1],
            fructose = df40_8.Fructose[1], lactate = df40_8.Lactate[1],
            acetate = df40_8.Acetate[2], propionate = df40_8.Propionate[1],
            ethanol = df40_8.Ethanol[1], co2 = 0.0, hydrogen = 0.0, water
            ↪  = 750.0,
            pyruvate = 0.0, succinate = 0.0, acetaldehyde = 0.0)
init_mass40_8 = conc_to_mass(init_st40_8, v)
```

Then, we can define the compound reaction function that lists all the pathways we believe are occuring.

```
function mixed_culture_fermentation(st; gluc_goal = (; glucose = 0.0),
↪  suc_goal = (; sucrose = 0.0), lact_cons_goal = (; lactate = 0.0),
↪  fruc_goal = (; fructose = 0.0), pyr_goal = (; pyruvate = 0.0),
↪  acet_amount = 0.5, lact_amount = 0.5, het_amount = 1.0, eth_amount =
↪  0.0, feed_oxygen = 0.0)
    # Sucrose is hydrolyzed
```

```
suc_st = sucrose_hydrolysis(st, goal = suc_goal)

# Glucose goes into either glycolysis or heterolactic fermentation
het_goal = (; glucose = (suc_st.glucose - (suc_st.glucose -
↪  gluc_goal.glucose)*het_amount))
glyc_goal = (; glucose = (suc_st.glucose - (suc_st.glucose -
↪  gluc_goal.glucose)*(1-het_amount)))
het_st = ethanol_heterolactate(suc_st, goal = het_goal)
glyc_st = glycolysis(suc_st, goal = glyc_goal)

pyr_st = merge(suc_st,
               (glucose = gluc_goal.glucose,
                ethanol = het_st.ethanol,
                lactate = het_st.lactate,
                co2 = het_st.co2,
                pyruvate = het_st.pyruvate + glyc_st.pyruvate -
                ↪  suc_st.pyruvate,
                hydrogen = het_st.hydrogen + glyc_st.hydrogen -
                ↪  suc_st.hydrogen))

# Fructose is also hydrolyzed
fruc_st = fructolysis(pyr_st, goal = fruc_goal)

# Check if there is any oxygen in the reactor and if there is
# consume the according pyruvate
ox_st = merge(fruc_st, (; oxygen = feed_oxygen))
new_st = aerobic_pyruvate_oxidation(ox_st)

# Then, pyruvate is converted to the various products
aceteth_amount = 1 - lact_amount - acet_amount - eth_amount
acet_goal = (; pyruvate = (new_st.pyruvate - (new_st.pyruvate -
↪  pyr_goal.pyruvate)*acet_amount))
lact_prod_goal = (; pyruvate = (new_st.pyruvate - (new_st.pyruvate -
↪  pyr_goal.pyruvate)*lact_amount))
aceteth_goal = (; pyruvate = (new_st.pyruvate - (new_st.pyruvate -
↪  pyr_goal.pyruvate)*aceteth_amount))
eth_goal = (; pyruvate = (new_st.pyruvate - (new_st.pyruvate -
↪  pyr_goal.pyruvate)*eth_amount))

aceteth_st = acetate_ethanol_fermentation(new_st, goal =aceteth_goal)
acet_st = pyruv_to_acetate(new_st, goal = acet_goal)
lact_st = pyruv_to_lact(new_st, goal = lact_prod_goal)
eth_st = pyruv_to_ethanol(new_st, pyr_goal = eth_goal)

prod_st = merge(new_st,
               (pyruvate = pyr_goal.pyruvate,
                acetate = acet_st.acetate + aceteth_st.acetate -
                ↪  new_st.acetate,
                    ethanol = aceteth_st.ethanol + eth_st.ethanol -
                    ↪  new_st.ethanol,
                    lactate = lact_st.lactate,
                    co2 = acet_st.co2 + aceteth_st.co2 + eth_st.co2 -
                    ↪  2new_st.co2,
                    hydrogen = acet_st.hydrogen + aceteth_st.hydrogen +
                    ↪  lact_st.hydrogen - 2new_st.hydrogen))

    prop_st = lact_to_propionate(prod_st, goal = lact_cons_goal)
end
```

9.1 Finding the true metabolic pathway

The above function is very useful. Given all its required parameters, it returns an output state of what it believes is the effluent concentrations. What we want is to find the parameter set for which the function output matches the experimental data. This is classic parameter estimation so if formulated correctly, we can very easily solve it with the SciML tooling. First, we need a function y = f(x,p) which given an input and a parameter set, returns an output. However, we want to have knowledge of the initial and final states, so we first a generalized function that does this work and while testing we can make it more specific to have the above signature. The next few code blocks are the backend which the optimization interface will use and for that reason it will be tangled to a src file and not to scripts. The function `mixed_culture_fermentation` defined above is necessary for this to work so it will also be tangled to that file.

```
<<final_pathway>>

function mixed_culture_optimization(init_st, final_st, p, v)
    init_mass = conc_to_mass(init_st, v)
    final_mass = final_st.*v
    mixed_culture_fermentation(init_mass, suc_goal = (; sucrose =
    ↪  final_mass[1]), gluc_goal = (; glucose = final_mass[2]), fruc_goal
    ↪  = (; fructose = final_mass[3]), lact_cons_goal = (; lactate =
    ↪  final_mass[4]), het_amount = p[1], acet_amount = p[2], lact_amount
    ↪  = p[3], eth_amount = p[4], feed_oxygen = p[5])
end
```

This function is a generalized function that does this exact thing. It takes an initial state that needs to be of the format `mixed_culture_fermentation` will accept and two vectors. The first is the final state which we want to get. It will be a vector of 7 numbers with the things HPLC gives. The first 4 are given as arguments, while the other 3 will be the variables we optimize for. The other vector is a vector of 5 elements containing our parameters. The function that we will use in the final optimization will not have final_{st} as a parameter, but rather we will make another function that calls it with a pregiven vector. In the same logic, we can write our loss function.

```
function fermentation_loss(init_st, final_st, p, v)
```

```julia
    model = mixed_culture_optimization(init_st, final_st, p, v)

    model_mass = [model.acetate, model.propionate, model.ethanol]
    exp_mass = final_st[5:7].*v
    sum(abs2, (model_mass .- exp_mass))
end
```

It takes the 4 arguments of the above function, runs it and compares it with the 3 unused variables of the output state.

Lastly, we want a predictor function for use after optimization. The optimization problem simply returns the parameter vector, however, what we are interested in is what the output state looks like and its loss function. This predictor function returns these 2 elements with the same signature as above.

```julia
function mixed_culture_predictor(init_st, final_st, p, v)
    mass_st = mixed_culture_optimization(init_st, final_st, p, v)
    conc_st = mass_to_conc(mass_st, v)

    loss = fermentation_loss(init_st, final_st, p, v)
    println("For the parameter set ", p, " we get the effluent \n",
     ↪  conc_st, " and a loss of ", loss, " with the experimental data.")
end
```

Now, we can test this system for a specific dataset. To do this, we will move back to the metabolic$_{\text{tests.jl}}$ file, but include the optimization interface source.

```julia
include(srcdir("metabolic_pathways", "opt_interface.jl"))

# Define the final state that we want to use.
final_st35_0 = Vector(df35_0[4, 2:7])
push!(final_st35_0, df35_0[3,8])

# Define the specialized loss function with signature f(u, p) which is
# what the Optimization interface accepts and also a predictor
# function which is also specialized and helps us with testing.
loss_35_0(u, p) = fermentation_loss(init_st35_0, final_st35_0, u, 0.8)
predictor_35_0(u) = mixed_culture_predictor(init_st35_0, final_st35_0, u,
↪  0.8)

using Optimization, OptimizationOptimJL

# u0 can be taken basically randomly and if its not too bad, the
# algorithm will converge
u0 = [0.85, 0.01, 0.1, 0.2, 1.5]

# The bounds are necessary as outside of some bounds,
# mixed_culture_fermentation will return an error due to finding an
# unfeasible pathway. We only want to search within the range of
# feasibility.
lbound = [0.8, 0.0, 0.00, 0.0, 0.0]
ubound = [1.0, 0.3, 0.3, 0.3, 1.6]
adtype = Optimization.AutoForwardDiff()

# Define the system both with and without AD to make sure AD is
# working as intended
optf = Optimization.OptimizationFunction(loss_35_0)
optf_ad = Optimization.OptimizationFunction(loss_35_0, adtype)

optprob = Optimization.OptimizationProblem(optf, u0, lb = lbound, ub =
↪  ubound)
optprob_ad = Optimization.OptimizationProblem(optf_ad, u0, lb = lbound, ub
↪  = ubound)

sol = solve(optprob_ad, Optim.BFGS())
```