`microPop`: Modelling microbial populations and communities in R

Helen Kettle
Email: Helen.Kettle@bioss.ac.uk

Biomathematics and Statistics Scotland (BioSS),
Kings Buildings, Edinburgh, EH9 3FD, UK.

August 3, 2017

Summary

1. `microPop` is an R-package for simulating the deterministic dynamics and interactions of microbial populations by solving a system of ordinary differential equations.

2. `microPop` contains data frames for a number of microbial functional groups (defined by their metabolic pathways) and these may be added to, or modified, by the user.

3. Default functions for rates of microbial growth, resource uptake, metabolite production and the entry and exit rates into and out of the system of each state variable, are provided but can be modified or replaced by the user.

4. `microPop` can be used to simulate growth in a single compartment (e.g. bio-reactor) or 'compartments' in series (e.g. human colon) or in a simple 1-d application (e.g. phytoplankton in a water column).

5. A microbial functional group can contain multiple strains in order to study adaptation and diversity. These strains have the same metabolic pathways but their parameter values may differ.

6. Simple interactions between viruses (bacteriophages) and bacteria can also be included in `microPop`.

7. In addition to this vignette there is (or will soon be!) a paper in Methods in Ecology and Evolution (in press). Please cite the MEE paper in any publications using `microPop`.

1 Introduction

`microPop` is an R package which uses ordinary differential equations (ODEs) to predict the dynamics and interactions of microbial communities. For example,

the general equations for rates of change of a microbial functional group (MFG), with quantity X, growing on a resource, with quantity R, at time, t, are given by,

$$\frac{dX(t)}{dt} = V_X^{in}(t)X(t) + G(t)X(t) - V_X^{out}(t)X(t) \quad (1)$$

$$\frac{dR(t)}{dt} = V_R^{in}(t)R(t) - \frac{G(t)X(t)}{Y} - V_R^{out}(t)R(t) \quad (2)$$

where V_i^{in} and V_i^{out} are the inflow and outflow rates of the system (units of inverse time) for microbes ($i = X$) and resources ($i = R$). $G(t)$ is the specific growth rate of microbes on the resource (also units of inverse time) and can be modelled in a variety of ways (see Appendix A). The second term in Eq. 2 is the uptake rate of the resource due to microbial growth where Y is the yield i.e. the quantity of microbial growth per unit of resource taken up. When there are multiple resources and several microbial groups with multiple strains then Eqs. 1 and 2 expand into a large system with multiple metabolic pathways. This is where `microPop` is a useful tool. Rather than coding these equations, the user simply gives a description of the system (using data frames, '`resourceSysInfo`' and '`microbeSysInfo`') and a data frame for each MFG and these equations are constructed and solved by the function **microPopModel** using the ODE solvers provided by the deSolve package (Soetaert et al., 2010). The output from this function is a list containing two elements - one is the solution to the ODEs i.e. a matrix of the values of the state variables over time (this is called '`solution`') and the other is a list containing all of the information used to produce the solution (called '`parms`').

Details of a number of MFGs found in the human large intestine (e.g. Bacteroides, Acetogens, Methanogens, Butyrate Producers, Lactate Producers and so on) as described by Kettle et al. (2015); and those found in the rumen (described by Munoz-Tamayo et al. (2016)) are included as data frames in the package and if the user simply wishes to use these MFGs then microPop can be used 'off the shelf'. However, if not, the user can add in any number of other MFGs (by providing a data frame in the correct format or by using a csv file and the R function **data**). Alternatively the user can simply modify the entries in the data frames of the MFGs provided to try out different parameter values etc (information on the format of these data frames is found using **help(MFG)** once the microPop library has been loaded in R).

Since microbial growth, resource uptake and production may be modelled in a number of ways, the choices behind microPop's default growth and uptake functions are explained fully in the Appendix. In brief the default functions assume that hydrolysis of polymers is bound up with microbial growth. However, it is also possible to model hydrolysis separately within microPop with some adjustments (e.g. see Section 3.2 on the rumen). Growth on multiple substrates may also be substitutable or essential (this can be defined in microPop within the '`Rtype`' category in the MFG data frames) and the equations used in these two cases are described in the Appendix. Growth on essential substrates can be completely determined by stoichiometric ratios such as those used by Munoz-Tamayo et al. (2016) in which microbes are included in the stoichiometries (again see Section 3.2 rumen example). If microbial biomass is not included in the stoichiometries, an estimation for the amount of each metabolic product can be made by subtracting the mass of the new microbial growth from the total

substrate uptake and then computing metabolite production using the stoichiometric ratios (the method used by Kettle et al. (2015)). Errors introduced by using this method are briefly discussed in Appendix C.

If water is required for microbial growth or is a by-product of growth then it should be included in the group's data frame (with '`Rtype`' of 'Sw') . It will not be used in growth calculations as it is assumed that water will not be limiting. However, when computing the mass available for products (if microbial biomass is not included in the stoichiometry) then the uptake of water or the production of water is needed to balance mass correctly (this is done by using the stoichiometric ratios).

Given the ability to redefine the functions called by the ODE solver, `microPop` can be applied to a large number of different microbial ecosystems. Although very complex ecosystems with multiple microbial groups and strains may be slow to run in R, we hope that the transparency and flexibility of the code and its accessibility will enable researchers to simulate fairly complex systems without taking on a large computing project.

2 Running `microPop`

After installing the microPop package, the function **microPopModel** is used to run the simulation. The user need only specify 4 of the input arguments (the others have defaults) these 4 are:

- microbeNames - a vector of the names of the microbial groups in your system, e.g. c('Bacteroides','Methanogens').
- times - a vector defining the time sequence at which output is required, e.g. seq(0,10,0.1).
- resourceSysInfo - this is a data frame or the name of a csv file describing the inflow, outflow, start values and molar masses of the substrates and products associated with the microbial groups specified in microbeNames. Use **help(resourceSysInfo)** for details.
- microbeSysInfo - this is a data frame or the name of a csv file describing the inflow, outflow and start values of the microbial groups specified in microbeNames. Use **help(microbeSysInfo)** for details.

Details of all the input arguments can be found via **help(microPopModel)**. One of the most important input arguments is '`rateFuncs`'. MicroPop has been designed to let the user modify/replace every function pertinent to modelling growth, uptake, production, pH dependency etc required to solve the ODEs. These functions are gathered together in a list and this list is entered via '`rateFuncs`'. The default versions of these functions (Table 2) are gathered together in a list called 'rateFuncsDefault' and the default input argument is '`rateFuncs=rateFuncsDefault`'. If the user wishes to define a new version of a function in this list, e.g. function X (where is X is any of the functions named in the top section of Table 2) then this can be done by defining a new list and a changing function X within that list e.g.

'`myRateFuncs=rateFuncsDefault`'
'`myRateFuncs$X=function()`'

Table 1: Microbial functional groups (MFGs) included in microPop. The first ten groups are based on those described by Kettle et al. (2015), the last three (below dividing line) are based on those described by Munoz-Tamayo et al. (2016). To see these data frames simply type in the group name at the R prompt. Users should be aware that the parameter values given in these data frames will almost certainly change with increasing knowledge of gut microbiota and in some cases are simply a 'best guess'.

MFG	Description	Examples
Bacteroides	Acetate-propionate-succinate group	Bacteroides spp.
NoButyStarchDeg	Non-butyrate-forming starch degraders	Ruminococcaceae related to Ruminococcus bromii. Might also include certain Lachnospiraceae.
NoButyFibreDeg	Non-butyrate-forming fibre degraders	Ruminococcaceae related to Ruminococcus albus, Ruminococcus flavefaciens. Might also include certain Lachnospiraceae.
LactateProducers	Lactate producers	Actinobacteria, especially Bifidobacterium spp, Collinsella aerofaciens
ButyrateProducers1	Butyrate Producers	Lachnospiraceae related to Eubacterium rectale, Roseburia spp.
ButyrateProducers2	Butyrate Producers	Certain Ruminococcaceae, in particular Faecalibacterium prausnitzii
PropionateProducers	Propionate producers	Veillonellaceae e.g. Veillonella spp., Megasphaera elsdenii
ButyrateProducers3	Butyrate Producers	Lachnospiraceae related to Eubacterium hallii, Anaerostipes spp.
Acetogens	Acetate Producers	Certain Lachnospiraceae, e.g. Blautia hydrogenotrophica
Methanogens	Methanogenic archaea	Methanobrevibacter smithii
Xsu	Sugar utilizers	
Xaa	Amino acid utilizers	
Xh2	Hydrogen utilizers	Methanobrevibacter smithii

Table 2: Top section of table: Functions contained in the list rateFuncsDefault (further details on these functions are included in the Appendix). Bottom section of table: other functions in microPop. To get help on the inputs and outputs of these functions use **help(functionName)** in R using the function names below.

Function name	Description
entryRateFunc	Rate of entry of each state variable to system at time t
removalRateFunc	Rate of exit of each state variable from system at time t
pHFunc	pH value at time t
pHLimFunc	pH limit on growth (varies between 0 and 1 for a given pH value)
extraGrowthLimFunc	Another limit on growth (default value is 1 i.e. no limit). This is included to allow the user to add in any kind of growth limitation as its output is used to scale the maxGrowthRate value)
growthLimFunc	This scales the maximum growth value (value between 0 and 1)
combineGrowthLimFunc	Combining growth on multiple resources
uptakeFunc	Uptake of resource due to microbial growth
productionFunc	Production of metabolites resulting from microbial growth
combinePathsFunc	Combining the results of growth on multiple metabolic pathways
createDF	Creates a data frame from a .csv file
derivsDefault	Describes the ODEs; called by **ode**
getGroupName	Returns the name of the group from the strain name
makeInflowFromSoln	Returns the exit rate of each state variable (matrix[time,variable])
microPopModel	Simulates growth of microbial populations (main function)
pHcentreOfMass	Finds the mean pH weighted by the pH limitation
plotTraitChange	Plots the average group trait over time (when there are multiple strains per group)
runMicroPopExample	Used to run the scripts for the examples described in Section 3

and then setting '`rateFuncs=myRateFuncs`' in **microPopModel**. More details on these rate functions are included in the Appendix. Furthermore, if the user wishes to make fundamental changes to the structure of microPop, the default function describing the ODEs, **derivsDefault**, may also be replaced via the input argument '`odeFunc`' in **microPopModel**.

MicroPop also contains a number of checking functions. These are controlled by the '`checkingOptions`' list input to **microPopModel**. These can be used to check that mass is conserved, that stoichiometries balance, that the solution does not become negative and so on - more information is given in the help for **microPopModel** under '`checkingOptions`'.

3 Example Applications

To look at the code used in these applications, find the location of the R scripts by typing **system.file(DemoFiles/ExampleFileName.R", package = microPop")** in R and then view the file in any text editor. The examples can be run by typing, for example, **runMicroPopExample('human1')** or by changing the directory in R (use **setwd**) to the folder containing the R scripts and using **source**. Most of the plots shown in this paper are automatically generated by microPop and can be tweaked using the plotOptions input list in **microPopModel**.

3.1 Modelling human gut microbiota

The model described by Kettle et al. (2015) uses 10 different microbial groups to represent the microbial community in the human colon (Table 1). Here we use three of these – Bacteroides, NoButyStarchDeg (starch degraders that do not produce butyrate) and Acetogens – to demonstrate some features of microPop. The names of the R scripts for each of these is in brackets in the section heading. The information describing the inflows and outflows of each state variable for these scenarios is contained in the data frames resourceSysInfoHuman and microbeSysInfoHuman which are included with the package and are based on the system described by Kettle et al. (2015) and Walker et al. (2005). To look at these simply type resourceSysInfoHuman or microbeSysInfoHuman at the R prompt. Since these contain information on all 10 groups used in the full simulation by Kettle et al. (2015) the reader can use these to look at the behaviour of any/all of the groups by changing the names in the input argument microbeNames in **microPopModel**.

3.1.1 Microbial growth in a constant environment [`human1.R`]

A simple example to show how microPopModel can be run using most of the default settings. In this scenario there is no limit on growth due to pH and Bacteroides dominate the system (Fig. 1)

3.1.2 Microbial growth with pH change [`human2.R`]

In this scenario, pH changes from 5.5 to 6.5 halfway through the simulation. This is implemented by altering **pHFunc** and turning on the pH limitation on growth (**microPopModel** input argument '`pHLimit`'). Due to their preferred

pH ranges (determined by pHcorners in the data frames for each group) NoButyStarchDeg now dominate the first half of the simulation, however when the pH rises to 6.5 Bacteroides regain dominance (Fig. 2).

3.1.3 Microbial growth in two compartments with downstream flow [human3.R]

In this scenario we simulate the pH variation along the human colon by defining the system as two compartments where the first one flows into the second. The pH is now constant in time but varies between compartments with the first one set at pH 5.5 and the second at 6.0. To simulate two compartments we add a loop to call **microPopModel** twice. The first call simulates growth in the first compartment over the whole of the simulation time. The results from this are then used to provide the entry rates to the second compartment in the second **microPopModel** call. The entry rates of each state variable are computed using the function **makeInputFromSoln** to create a matrix called inflow.mat which is used in the newly defined **entryRateFunc**. The starting conditions are the same in each compartment but can be changed by using different sysInfo files for each call to **microPopModel**. The results (Fig. 3) show that NoButyStarchDeg dominate in first compartment (top row) and Bacteroides begin dominating the second compartment but this changes due to large inflow of NoButyStarchDeg from the previous compartment.

3.1.4 Microbial growth with pH change and multiple strains per group [human4.R]

In this scenario we include microbial diversity by assigning 5 strains to each microbial group (**microPopModel** input argument, '`numStrains=5`'). We assume that the strains within a microbial group have the same metabolic pathways i.e. those specified in the group data frame, but diversity is incoporated by randomly varying some of the strain parameters (based on Kettle et al. (2015)). The extent of this variation is controlled by setting '`percentTraitRange`' and '`maxPHshift`' in the '`strainOptions`' input list. The parameters that can be randomly varied between strains are halfSat, yield, maxGrowthRate and pHtrait but a subset of these can be defined using the **microPopModel** input argument '`strainOptions$randomParams`' (the default is all four). Furthermore, trait values may be traded-off against each other to avoid the creation of a 'super bug' using '`strainOptions$applyTradeOffs=TRUE`' and setting the chosen parameters in '`strainOptions$tradeOffParams`' (only two parameters are allowed and pH trait is not one of them since the fitness of the pH trait can change from 'good' to 'bad' as the environment changes). Moreover, the user may also specify the parameter values for individual strains using '`strainOptions$paramsSpecified=TRUE`' and giving the name of the csv file or the data frame object where these values are stored in '`strainOptions$paramDataName`'. This file/data frame has the following column headings: strainName, paramName, paramVal, paramUnit, resource, path. Not all parameter values need to be specified - those that are specified will simply overwrite the randomly generated values. Fig. 4a,b show the results for each strain (to plot the sum of the strains in each group set '`plotOptions$sumOverStrains = TRUE`').

When there are multiple strains per group it is possible to examine how a

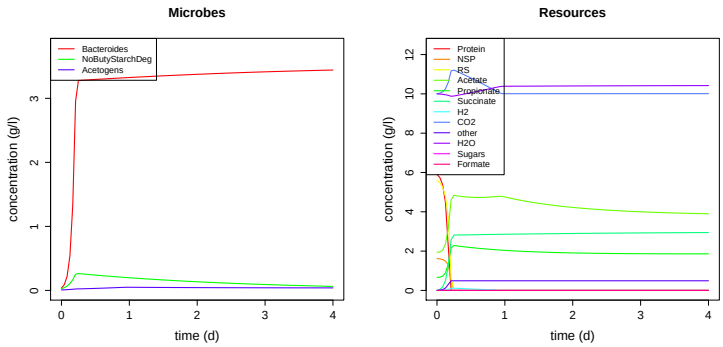

Figure 1: Human Colon Application (human1)

mean group trait adapts over time using a biomass-weighted average at each time step:

$$\overline{x(t)} = \frac{\sum_i^n x_i m_i(t)}{\sum_i^n m_i(t)} \tag{3}$$

where $\overline{x(t)}$ is the average group trait at time t, x_i is the trait value for strain i and $m_i(t)$ is the mass of strain i at time t. For example when pH changes, strains which prefer that pH will flourish whilst others will be washed out. The centre of mass of this limit function can be computed using the function **pHcentreOfMass** and we define this one parameter as the pH trait. We can compute and plot the change in time of any of the stochastically-varying parameters/traits using the function **plotTraitChange**. Fig. 4c shows the variation of the pH trait over time for each microbial group. If microPop is run multiple times with populations of strains with different random traits each time (this is done by changing the value of '`strainOptions$seed`' each time) and run to steady state then a population of viable microbial communities can be generated (e.g. Kettle et al. (2015)). Moreover, if only one strain per group is specified it is also possible to randomly generate its parameters for each run by setting **microPopModel** input argument '`oneStrainRandomParams=TRUE`'. This might be useful, for example, for generating model output to represent samples from a number of volunteers.

3.2 Methane production from rumen microbiota [`rumen.R`]

In this example we show how microbial biomass can be included within the stoichiometries and how hydrolysis can be included. The model we use here is based on that described by Munoz-Tamayo et al. (2016), however, we simplify this for demonstration purposes by considering only constituents dissolved in the rumen fluid (thereby removing gas transfer from the fluid fluid to the rumen head space), removing carbon chemistry (we only consider dissolved inorganic carbon) and removing the mechanistic calculation of pH from acid-base reactions. Also, we use units of mass rather than moles. We follow the convention of Munoz-Tamayo et al. (2016) where dissolved components are denoted S_i, polymer components are denoted Z_i and microbial groups are denoted X_k.

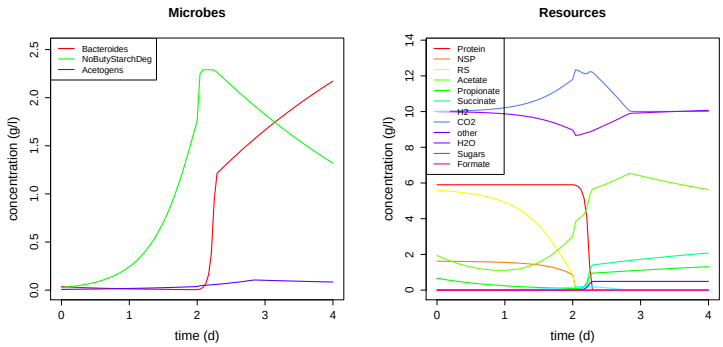

Figure 2: Human Colon Application (human2) - pH change.

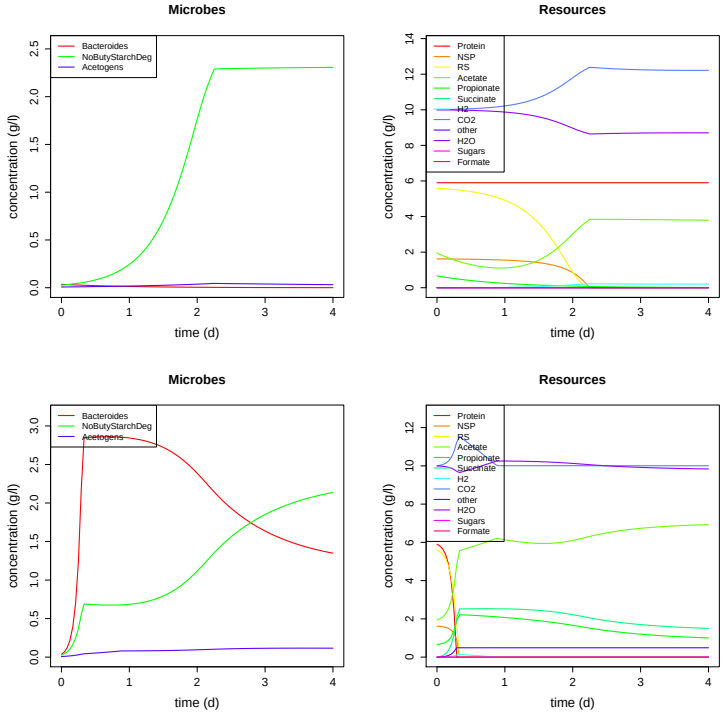

Figure 3: Human Colon Application (human3) - two compartments (top row: first compartment, bottom row: second compartment).

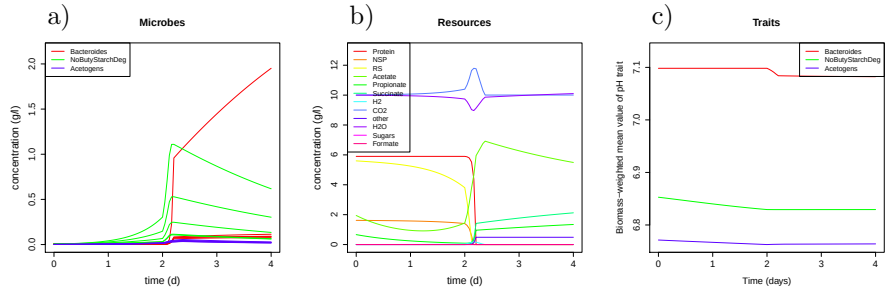

Figure 4: Human Colon Application (human4) - five strains per group.

The system consists of polymers which are hydrolised to soluble sugars and amino acids. The microbial groups are sugar-utilisers (X_{su}), amino-acid utilisers (X_{aa}) and hydrogen utilisers (X_{h2}); the data frames for these three groups are provided as part of microPop (type '`Xsu`', '`Xaa`' or '`Xh2`' at the R prompt to view them). Also in this system dead microbial cells are recycled into the polymers. Fig. 5 shows a schematic diagram of the system and notation of state variables (figure caption).

The metabolic pathways for these three groups differ from the those of the 10 groups we used in the human colon model in that their stoichiometries contain microbial biomass. Note that microbial products are referred to as 'Biomass' in the group file with '`Rtype`' of 'P_b' for biomass product. Also note that it is not necessary to add 'Biomass' to the resource SysInfo file as these information is covered in the microbe sysInfo file. Furthermore for all three groups every substrate is essential. A 'key resource' must be defined and the biomass yield upon this (mass of biomass per mass of key resource consumed) is given for this resource only. More detail on how this is modelled is given in the Appendix.

To include hydrolysis of polymers we first need to add polymers as state variables. Since none of the microbial groups use polymers directly they are not included in any of the group data frames which means they are not automatically included as state variables by **microPopModel**. Thus we add Z_{nsc}, Z_{ndf} and Z_{pro} to the microPop data frame for X_{su}, e.g. Xsu[['Zndf']]=c('X', rep(NA,6)). We then add in the parameters needed to model hydrolysis and recycling of dead cells into polymers as these are not included in the input files.

The next step is adding the process of hydrolysis to the system. This involves the break down of polymers and the production of S_{su} and S_{aa}, therefore, the **removalRateFunc** now includes the reduction rate for polymers and the **entryRateFunc** includes the equivalent increase for S_{su} and S_{aa}. Similarly the death of microbial cells is included in **removalRateFunc** and the increase in polymers from the dead cells is included in **entryRateFunc**.

Using the same settings as Munoz-Tamayo et al. (2016), we now investigate how increasing the initial concentrations of the feed polymers, Z_{nsc}, Z_{ndf} and Z_{pro}, affects the concentration of methane in the rumen (S_{ch4}). Thus we set the initial polymer concentrations at 1 g/l and then increase each one in turn to 20 g/l (Fig. 6). Increasing Z_{ndf} and Z_{pro} leads to increasing methane concentrations as expected, however, the second row in Fig. 6 shows that somewhat counter-intuitively the amount of methane produced decreases as initial concentrations of Z_{nsc} increases over a threshold between 15-20 g/l. SIC and S_{h_2} (not

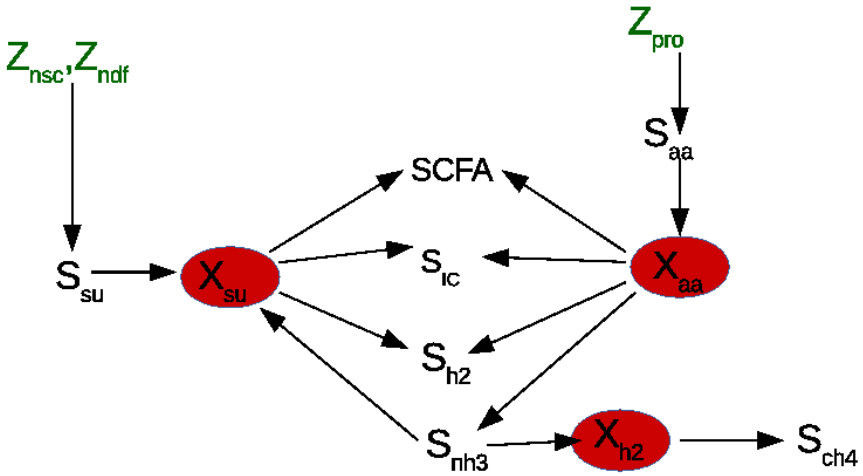

Figure 5: The rumen system based on the model by Munoz-Tamayo et al. (2016) consists of polymers (non-structural carbohydrates (Z_{nsc}), cell wall carbohydrates (Z_{ndf}) and proteins (Z_{pro}) which are hydrolised to the soluble components: sugars (S_{su}) and amino acids (S_{aa}). The microbial groups are sugar-utilisers (X_{su}), amino-acid utilisers (X_{aa}) and hydrogen utilisers (X_{h2}). They convert their respective substrates to short chain fatty acids, SCFA, (acetate, S_{ac}, butyrate S_{bu} and propionate S_{pr}), hydrogen (S_{h2}), ammonia (S_{nh3}), inorganic carbon (S_{IC}) and methane (S_{ch4}). Dead microbial cells are recycled to the polymer compartments (not shown).

shown) both increase with Z_{nsc}, therefore the cause of this appears to be the decrease in S_{nh3} (third column in Fig. 6) which rapidly falls to zero for high initial values of Z_{nsc}. This is because Z_{nsc} is hyrolysed at a much faster rate (0.2 h^{-1}) than Z_{ndf} (0.05 h^{-1}) so increased Z_{nsc} leads to increased S_{su} and rapid growth of X_{su} and hence rapid uptake of S_{nh3}. The depletion of S_{nh3} inhibits the growth of X_{h2} and thus the production of methane in this simple model example.

3.3 Nutrient-light balance for phytoplankton growth in the water column [`phyto.R`]

This example shows how microPop can be used in a 1D application to examine the depth at which phytoplankton blooms occur given they rely on nutrients welling up from below and sun light entering from above. By simulating the competing growth of three different (theoretical) microbial groups we show how the phytoplankton form a vertical assemblage based on their different levels of requirement for light and nutrient. We begin by simulating growth of just one phytoplankton group for 3 months over a depth of 20 m divided up into 1 m layers (when running **runMicroPopExample('phyto')** you will be prompted to enter a case number - this is case 1). At the beginning the phytoplankton are spread evenly through the depth of the water column (for example this

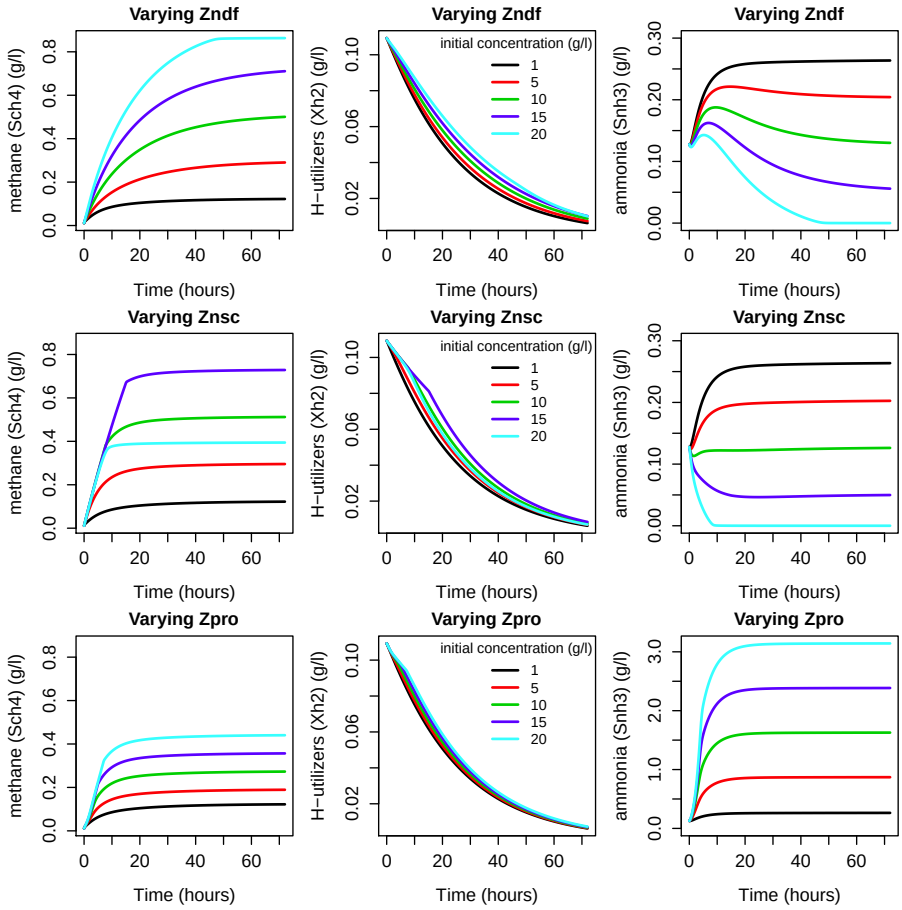

Figure 6: Methane concentration in the rumen for initial concentrations between 1 and 20 g/l (legend in centre column) of the feed polymers Z_{nsc}, Z_{ndf} and Z_{pro} (while one polymer concentration is changed the other two are held at 1 g/l). Note change in scale for S_{nh3} for Z_{pro}.

may occur after vertical mixing caused by high winds). Thereafter there is no mixing and the phytoplankton are stationary in water but grow at different rates according to the light and nutrient levels at their particular depth. To define this system in microPop we consider nutrient to be the only resource since light is not depleted through microbial use. The limiting effect of light on growth is incorporated via the **extraGrowthLimFunc** as the output from this function is used to scale the maximum growth rate in a similar way to **pHLimFunc**. The light level is computed using

$$\exp(-k_L z) \tag{4}$$

where k_L is the light attenuation coefficient (we use k_L=0.5 m^{-1}) and z is depth (distance below surface; m). Nutrient upwelling is incorporated into **entryRateFunc** by assuming that the inflow of nutrient increases with depth such that

$$I_N = v_N z \tag{5}$$

where v_N is the inflow rate of nutrient with depth (g l^{-1} m^{-1}). There is no wash out rate for resources but we set a small wash out rate for the phytoplankton of 0.005 d^{-1} (see 'systemInfoMicrobesPhyto.csv') to represent death rate. Note all parameter values for the microbial groups are not based on data - this example is simply for illustration purposes.

The microPopModel function is then called to simulate growth over 3 months for each depth and the results are saved after each model run. Fig. 7a shows how the magnitude and depth of the bloom changes with time as nutrients are depleted when there is only one group present (Phyto1).

We now add in 2 more groups (choose case 2 when running **runMicroPopExample('phyto')**). The groups have different requirements for nutrient and light as determined by their half saturation values (K_N and K_L respectively - see Fig. 7 caption). All three groups start with the same concentration; Fig. 7b shows how over time the groups occupy different levels in the water column.

3.4 Bacteriophages [`phages.R`]

`microPop` is not really designed for the study of bacteriophages (viruses which attack bacteria) but we show how phages can be included in `microPop` in a simplistic way. In this example we consider 2 (fictious) groups of bacteria (called Bacteria1 and Bacteria2) and 2 bacteriophages called Virus1 and Virus2. Both bacteria have the same substrate, nutrient, and the same parameters with the difference that Bacteria2 has a higher maximum growth rate than Bacteria1. Virus1 attacks Bacteria1; Virus2 attacks Bacteria2. The two viruses have the same parameter values and differ only in their choice of host cell (bacterial group). We consider a simple system with a constant dilution rate of 0.1 d^{-1}. There is inflow of nutrient but nothing else and all variables have a starting value of 1.

In order to infect a host cell, the bacteriophage attaches itself to the bacterial cell wall and then injects its genetic material (its nucleic acid) into the host cell, switching the cell's programme in its favour so the host cell will eventually die and release about 100 new phage particles. To model this within microPop we make some simplifying assumptions. Firstly, since one phage attacks one

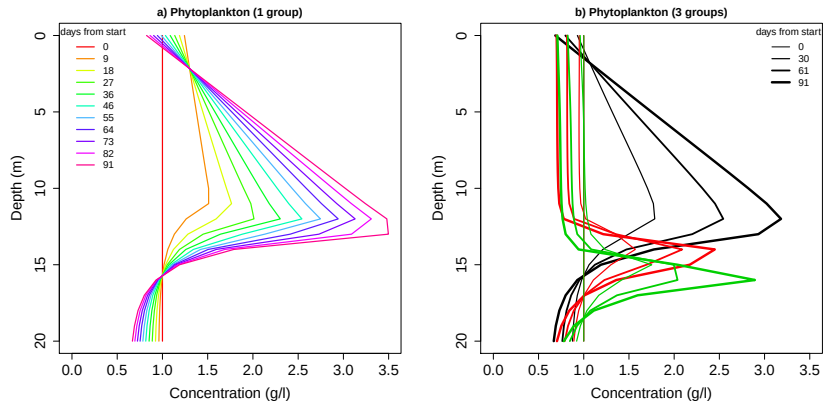

Figure 7: a) Concentration of Phyto1 at 9 day intervals when it is the only group present (K_N=1e-6 g l^{-1}, K_L=0.8 light units). b) Concentration of all three groups at monthly intervals with Phyto1 in black (K_N=1e-6 g l^{-1}, K_L=0.8 light units), Phyto2 in red (K_N=1e-4 g l^{-1}, K_L=0.4 light units) and Phyto3 in green (K_N=1e-2 g l^{-1}, K_L=0.2 light units).

bacterial cell, the 'consumption' rate does not follow a Monod Equation but it is more like a host-parasitoid model where the rate of change of the number of cells of the virus, V, due to viral attack on B bacterial cells is

$$\frac{dV}{dt} = \alpha V B \tag{6}$$

where α is the specific reproduction rate (number of new virus cells made from one viral cell per bacterial cell per day). To put this in microPop we put the maxGrowthRate of V1 on B1 equal to α and redefine **growthLimFunc** so the 'limitation' is now simply B rather than the Monod equation (this is multiplied by V later in **derivesDefault**). The rate of change of the number of bacterial cells due to death by virus attack is

$$\frac{dB}{dt} = -\frac{\alpha}{Y} V B \tag{7}$$

where Y is the yield i.e. the number of new virus cells per bacterial cell (note $\alpha = Yb$ where b is the binding rate (units of $V^{-1}d^{-1}$)). By using the equations above we have made the simplifying assumption that there is no time delay between viral attack and the production of new virus cells. To put in a time delay **derivsDefault** could be altered to use **dde** rather than **ode** from the package `deSolve` which microPop depends upon.

To add in mutations of Bacteria1 to a resistant strain we simply redefine **entryRateFunc** so that a fraction of the Bacteria1 population is converted to a resistant group ('resistantBacteria1', B_1^R) per day, denoted f_B. Thus the rate of change of B_1^R due to mutations is

$$\frac{dB_1^R}{dt} = f_B B_1 \tag{8}$$

and the loss rate from B_1 is the negative of this (also modelled via **entryRateFunc**).

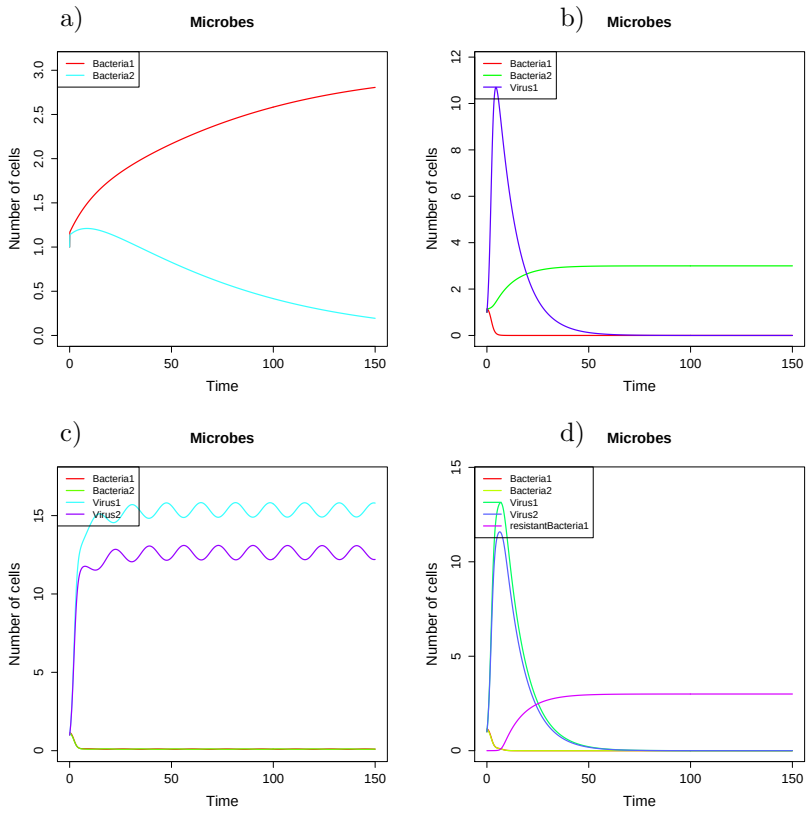

Figure 8: a) Case 1: Two bacterial groups compete for one substrate (Nutrient); no viruses present. b) Case 2: A virus (Virus1) which attacks Bacteria1 is added to the system. c) Case 3: As in b) but a virus (Virus2) which attacks Bacteria2 is also added to the system. d) Case 4: As in c) but Bacteria1 randomly mutates into a group which is identical to Bacteria1 apart from it has resistance to Virus1.

We run microPop for 4 different system scenarios (when running **runMicroPopExample('phages')** you will be prompted to choose from case 1 to 4); the results are shown in Fig. 8. To begin with we look at the system without viruses and see the two bacteria competing for nutrient, since Bacteria2 has the highest growth rate it dominates the system (case 1; Fig. 8a). We now add in Virus2 which attacks Bacteria2 allowing Bacteria1 to dominate the system causing Bacteria2, and hence Virus2, to die out (case 2; Fig. 8b). If we now add in Virus1, so that we have both bacterial groups and both viral groups, we see more complex dynamics emerge (case 3; Fig. 8c). In the fourth case we add in random mutations within the Bacteria1 group to a strain that is resistant to Virus1, called resistantBacteria1. This has all the same characteristics as Bacteria1 but is not present at the start of the simulation and is generated at a rate $f_B B_1(t)$. We set $f_B = 0.001$ d $^{-1}$ which means that 0.1% of the population of Bacteria1 mutates into the resistant strain per day (case 4; Fig. 8d).

Appendix

Most of the equations governing microbial growth in microPop are based on a previous model (Kettle et al. (2015) (open access)) for which a detailed description is given in the supporting information. In microPop the governing equations are contained in the function derivsDefault.R which is the function file for the ODE solver (**ode** from the `deSolve` pacakge). The functions called within derivsDefault.R are described in the sections below. These functions are collected into a list called rateFuncsDefault. This list is called by default for the input argument rateFuncs in **microPopModel** (also see Table 2 and Section 2 for details).

A Growth Equations

A microbe has a maximum specific growth rate, here denoted by μ^m (with units of inverse time), at which it may grow if there is no limitation on growth. Growth limitation functions, here denoted by λ, scale the maximum growth rate and their output must lie in the interval [0,1]. Limitations are typically due to substrate availability (e.g. λ_S) and pH, (λ_{pH}), however the user may add in further limitations using the function **rateFuncs$extraGrowthLimFunc**.

A.1 Substrate Limitation

For a substrate, i, with concentration, S, we express the limit on growth using the Monod equation

$$\lambda_{Se}(S_i) = \frac{S_i}{K_i + S_i}, \tag{9}$$

where K_i is the half-saturation constant (with the same units as those of S_i). This applies to growth on essential substrates ('`Rtype`' of 'Se'). However, if the microbe can grow on a variety of substrates then, for N_s substitutable substrates ('`Rtype`' of 'S'), the limit on growth due to substrate, i, is given by

$$\lambda_{Ss}(S_i) = \frac{S_i/K_i}{1 + \sum_{j=1}^{N_s} S_j/K_j} \tag{10}$$

based on Ballyk and Wolkowicz (1993). Note that if water is a resource ('`Rtype`' of 'Sw') then it is not included in growth limitation equations as we assume water is not limiting. These options are included in the default function in microPop but can be altered by redefining **rateFuncs$growthLimFunc**.

A.2 pH limitation

It is well known that pH has a significant effect on growth rates (Walker et al., 2005) and that this varies between different microbial groups. To incorporate this we define a pH limitation function for each group, denoted here by $\lambda_{pH}(g)$ for group g. The preferred pH range for each group is stated in the group's data frame under 'pHcorners' and the default function is a trapezium with the points on the x axis (pH) specified by pHcorners and the y-axis (limitation) values (0,1,1,0). However, this function can be redefined using **ratefuncs$pHLimFunc** to allow any form of pH limitation. The output from this

function, which must lie between 0 and 1, is used to scale the specific growth rate, μ, discussed in the previous section.

A.3 Combining growth on substrates

If there are N_s substrates they are combined in one of two ways. If the substrates are substitutable i.e. the microbes can grow on any of them, then, for group, g, we use,

$$\mu_s = \lambda_{pH}(g) \sum_{i=1}^{N_s} \mu_i^m \lambda_{Ss}(S_i). \tag{11}$$

If the substrates are essential i.e. there is no growth without all of them, then we use,

$$\mu_e = \lambda_{pH}(g) \mu^m \prod_{i=1}^{n} \lambda_{Se}(S_i), \tag{12}$$

where $\prod$ indicates multiplication of the following terms; note μ^m is not substrate specific. If the microbial group can grow on substitutable resources but also requires, or is boosted by, another resource, S_b, (which it can not grow on alone) then we define another growth limitation - that from the booster which is given by

$$\lambda_b = f_b + (1 - f_b)\lambda_{Se}(S_b) \tag{13}$$

where f_b is the fraction of the maximum growth achievable if the boosting resource is not present (i.e. when $\lambda_{Se}(S_b) = 0$). Growth rate is given by

$$\mu = \lambda_b \mu_s. \tag{14}$$

Note that if water is a resource ('`Rtype`' of 'Sw') then it is not included in growth equations as we assume water is not limiting. To alter any of these expressions redefine **rateFuncs$combineGrowthLimFunc**.

B Resource Uptake

The rate of uptake of a resource, i is given by the microbial mass growth rate on that resource divided by the mass yield of microbes, $Y^g_{X,i}$, on that resource, i.e. the mass of microbial growth resulting from the consumption of 1 g of the resource. We use the superscript g to distinguish it from molar yield. Thus if the microbial concentration is X then the biological uptake rate of a substitutable resource, S_i, is

$$\lambda_b \lambda_{pH}(g) \frac{\lambda_{S_s}(S_i)\mu_i^m}{Y^g_{X,S_i}} X. \tag{15}$$

If there are multiple essential resources, then the uptake rate of essential resource i must be proportional to the uptake of the other resources according to the stoichiometry. To ensure the stoichiometric mass ratios are maintained we choose one of the substrates to be the 'key resource' (denoted S_k), compute its uptake (shown in parentheses in the equation below) and then determine the uptake of the other substrates using

$$\frac{m_i n_i}{m_k n_k} \left(\lambda_{pH}(g) \mu^m \frac{\lambda_{S_e}(S_k)}{Y^g_{X,S_k}} X \right), \tag{16}$$

where the molar mass of i is denoted m_i and the number of moles of i in the stoichiometry is n_i. If the substrate is a boosting resource then we compute the uptake rate of the substitutable resources (shown in square brackets in the equation below) and then use the mean of the stoichiometric masses of the substitutable resources, such that

$$\frac{m_b n_b}{\frac{1}{n}\sum_{i=1}^{n}(m_i n_i)}\left[\lambda_b \lambda_{pH}(g)\sum_{i=1}^{n}\left(\frac{\lambda_{S_s}(S_i)\mu_i^m}{Y_{X,S_i}^g}\right)X\right] \tag{17}$$

Note that in the stoichiometries the substitutable resources are all regarded as hexose (or hexose equivalents) and therefore have the same m and n values. To alter any of these expressions redefine **rateFuncs$uptakeFunc**.

If water is needed for growth and is included in the stoichiometry in the group data frame then we also compute the uptake of water. This is done in **derivsDefault** and is calculated using mass stoichiometric ratios.

C Metabolite Production

When microbes grow on a substrate they release waste products (metabolites; '`Rtype`' of 'P') which in some cases can become substrates for other microbes. The rate of production of these metabolites can be computed in two ways. Firstly if the stoichiometry also contains biomass product (e.g. as in Munoz-Tamayo et al. (2016); '`Rtype`' of 'Pb') and if all of the substrates on the pathway are essential then the production rate is computed similarly to uptake rate such that the uptake rate of the key resource, U_k, is scaled by the mass stoichiometry ratio, e.g. for metabolic product, j, the production rate is,

$$\frac{m_j n_j}{m_k n_k} U_k. \tag{18}$$

However, if microbial growth is not included in the stoichiometry then, in order to conserve mass this must be substracted from total resource uptake and then the remaining mass is divided between the metabolites using stoichiometric ratios. If this case, if U_i is the uptake rate of substrate i, there are N_p metabolites, and N_s substrates and dX is the rate of microbial growth, then we estimate the production rate of metabolite j by

$$\frac{m_j n_j}{\sum_{k=1}^{N_p} m_k n_k}\left(\sum_{i=1}^{N_s} U_i + U_w - dX\right) \tag{19}$$

where U_w is the uptake rate of water if it is included in the group's stoichiometry. These expressions can be altered by redefining **rateFuncs$productionFunc**.

The first method requires microbial growth to be contained in the stoichiometry. This is usually done by assuming that the molecular formula for microbiota is $C_5H_7O_2N$ (Batstone et al., 2002) - the composition is given in detail in Table 3. Since this contains nitrogen, then the stoichiometries can no longer simply be carbohydrate based. Commonly ammonia is added to the stoichiometry to provide nitrogen. In our previous work (Kettle et al. (2015)) we do not include nitrogen but we assume that it is not limiting. Fortunately the errors introduced by using the approximation of Eq. 19, instead of stoichiometries which include microbial biomass, are very small due the following:

Table 3: Composition of microbiota using the formula $C_5H_7O_2N$ (Batstone et al., 2002).

atom	molar mass (g)	number	mass (g)	percentage mass (%)
C	12	5	60	53.1
O	16	2	32	28.3
N	14	1	14	12.4
H	1	7	7	6.2

- The values we use for the biomass yield on a substrate implicitly includes ammonia whether or not it is included in the stoichiometries. This also goes for other elements e.g. minerals etc that are not explicitly included.
- If ammonia (NH_3) and biomass are included in the stoichiometry and the molecular formula for biomass is $C_5H_7O_2N$ (Batstone et al., 2002) then due to conservation of N, for 1 mole of biomass growth, 1 mole of ammonia is taken up and all of the N in the ammonia must go to biomass creation (not metabolite production). This means that the maximium mass of ammonia that can go to products for 1 mole of biomass growth is 3 g (i.e. H_3). Given 1 mole of biomass is 113 g (Batstone et al., 2002) and since mass yields (mass biomass/mass substrate) are typically well below 1 e.g. on hexose it is approx. 1/3, then substrate mass taken up for 1 mole of microbial growth, is 339 g, leaving 226 g for product formation - thus the addition of 3 g from ammonia is negligible (approx. 1%).

D Growth on multiple paths

The previous sections all describe processes occuring on one metabolic pathway. However, some microbial groups will have multiple metabolic pathways. If this is the case then the total microbial growth for the group is a result of growth on each of these pathways. However, more metabolic pathways should not immediately mean more growth therefore we do not simply add the growth rates. Furthermore we also do not simply assume that the microbiota will divide their time equally between each of the pathways. Instead we assume that the more potential growth there is on a pathway the more the microbiota will prioritise it. In reality it may be that the microbiota will only grow on the favourable pathway and not at all on the others but at present we scale the growth on each pathway according to its fraction of the total growth. For example for growth μ on three pathways the actual growth on pathway, μ_i, is

$$\frac{\mu_i}{\sum_{j=1}^{3}\mu_j}\mu_i. \tag{20}$$

This expression can be altered by redefining **rateFuncs$combinePathsFunc**. However, it should be noted that complete pathway switching (e.g. where an **if** statement determines which pathway to choose) can lead to rapid rate changes creating a stiff system of ODEs. This can cause the ODE solver to fail or at the very least become extremely slow (different solver methods may be specified using odeOptions in **microPopModel**).

Acknowledgements

Scottish Goverment's Rural and Environment Science and Analytical Services Division (RESAS).

References

Ballyk, M.M. and Wolkowicz, G.S.K. (1993). Exploitative competition in the chemostat for two perfectly substitutable resources. *Mathematical Biosciences*, 118:127–180.

Batstone, D.J., Keller, J., Angelidaki, I., Kalyuzhnyi, S.V., Pavlostathis, S.G., Rozzi, A., Sanders, W.T., Siegrist, H., and Vavilin, V.A. (2002). *Anaerobic Digestion Model No.1 (ADM1)*. IWA Publishing, London.

Kettle, H., Louis, P., Holtrop, G., Duncan, S.H., and Flint, H.J. (2015). Modelling the emergent dynamics and major metabolites of the human colonic microbiota. *Environmental Microbiology*, 17:1615–1630.

Munoz-Tamayo, R., Giger-Reverdin, S., and Sauvant, D. (2016). Mechanistic modelling of in vitro fermentation and methane production by rumen microbiota. *Animal Feed*, 220:1–21.

Soetaert, K., Petzoldt, T., and Woodrow Setzer, R. (2010). Solving differential equations in r: package desolve. *Journal of Statistical Software*, 33:1–25.

Walker, A.W., Duncan, S.H., McWilliam Leitch, E.C., Child, M.W., and Flint, H.J. (2005). pH and peptide can radically alter bacterial populations and short-chain fatty acid ratios within microbial communities from the human colon. *Applied Environmental Microbiology*, 71:3692–3700.