An introduction to the ABM model in R

1. Overview

The ABM model predicts conversion of animal manure or other (high-moisture) organic wastes to methane (CH_4) and carbon dioxide (CO_2) under anaerobic conditions. The name comes from **a**naerobic **b**iodegradation **m**odel. With multiple microbial groups and group-specific parameters describing kinetics and yield, the model can predict realistic short- and long-term responses to temperature change and other perturbations. Although it was storage of animal slurry (liquid manure) in unheated channels or tanks that drove the initial development of the model, with its flexibility it is well-suited to simulate CH_4 emission or biogas production from other organic waste under a range of conditions, including in anaerobic digesters, particularly in the presence of temperature variations. The purpose of this document is to demonstrate the use of the ABM R package, which is a flexible implementation of the model. For a detailed description of the model itself, see Dalby et al. (2020a, 2020b).

2. Installation

The ABM package is available on GitHub and so can be installed with the `install_github()` function from the devtools package, which must be installed first. These steps must be carried out once to install both packages:

```
install.packages('devtools')
devtools::install_github('sashahafner/ABM')
```

And to use the ABM model, the package must be loaded.

```
library(ABM)
```

3. A simple example: methane emission from stored slurry

By default, the `abm()` function simulates degradation of animal manure from a 33 m^3 storage tank or channel with a 30 day emptying interval. Fresh slurry is added continuously at a rate of 1000 kg d^{-1}, and when emptied a residual of 10% of the total manure mass is left in the storage. Default values are included for all arguments, including the first two, which set the length of the simulation (365 d) and the time interval in the output (1 d).

In this example, the model is used to predict dynamics of CH_4 emission, microbial biomass, and VFA accumulation. The following call runs the ABM model with default argument values.

```
out1 <- abm()
```

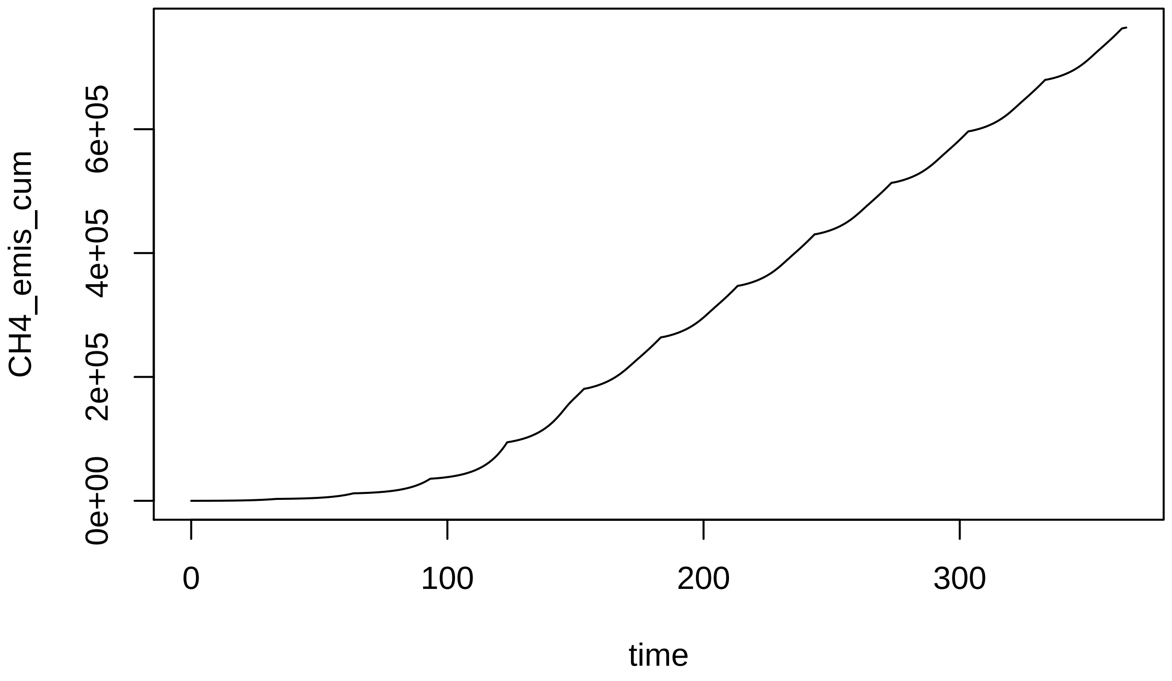
Output is, by default, a data frame with predicted variables over time (see Section X for alternatives). Typically the primary variable of interest is CH_4 emission, which is returned as a total (g) and rate (g/d), overall or normalized to COD or VS loading:

```
names(out1[grepl('^CH4', names(out1))])
```

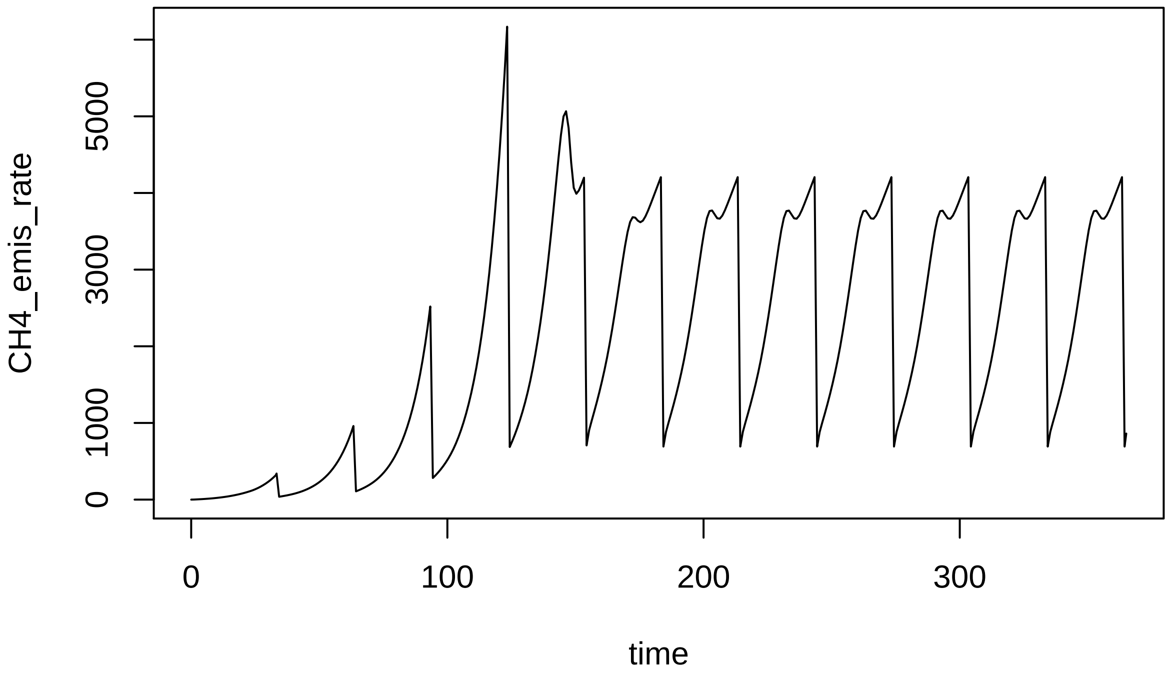
```
##  [1] "CH4_emis_cum"         "CH4_emis_rate"        "CH4_emis_rate_slurry"
##  [4] "CH4_flux"             "CH4_emis_rate_COD"    "CH4_emis_rate_dCOD"
##  [7] "CH4_emis_rate_VS"     "CH4_emis_cum_COD"     "CH4_emis_cum_dCOD"
## [10] "CH4_emis_cum_VS"
```

Total cumulative emission (g) and emission rate (g/d) are plotted below.

```r
plot(CH4_emis_cum ~ time, data = out1, type = 'l')
```

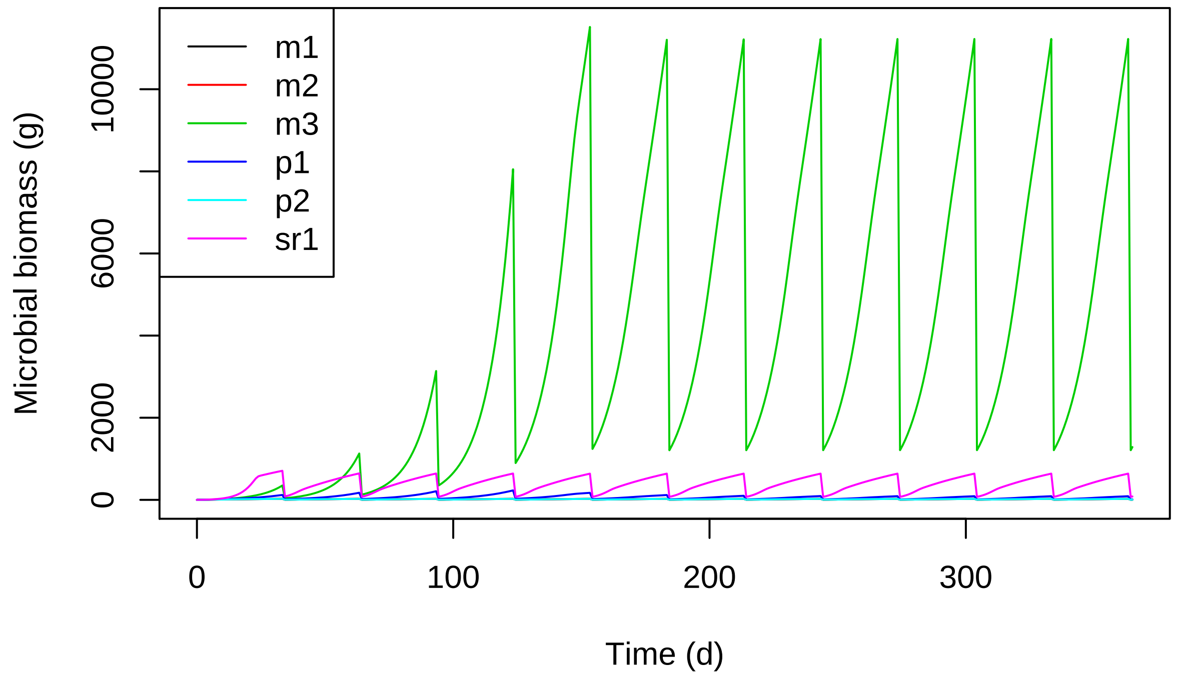


```r
plot(CH4_emis_rate ~ time, data = out1, type = 'l')
```



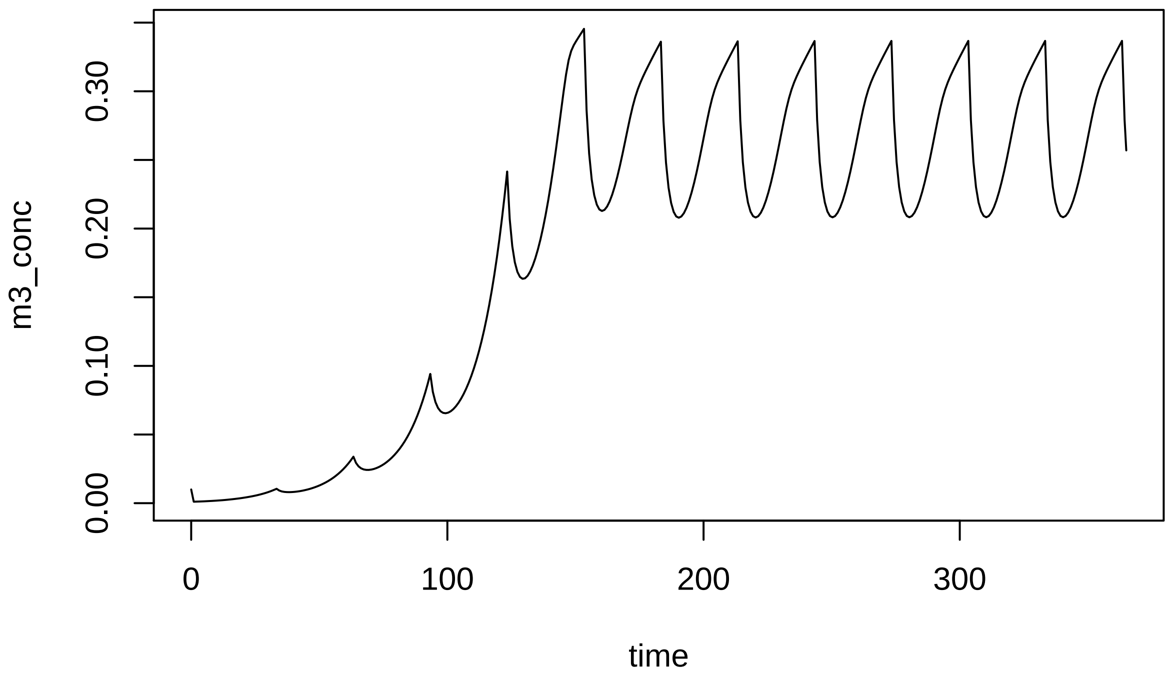
Microbial biomass (g) is given in columns with names that match those used for the names of the groups (defaults shown below, set within the `grp_pars` argument).

```r
matplot(out1$time, out1[, nn <- c('m1', 'm2', 'm3', 'p1', 'p2', 'sr1')],
        type = 'l', lty = 1, xlab = 'Time (d)', ylab = 'Microbial biomass (g)')
legend('topleft', nn, col = 1:6, lty = 1)
```



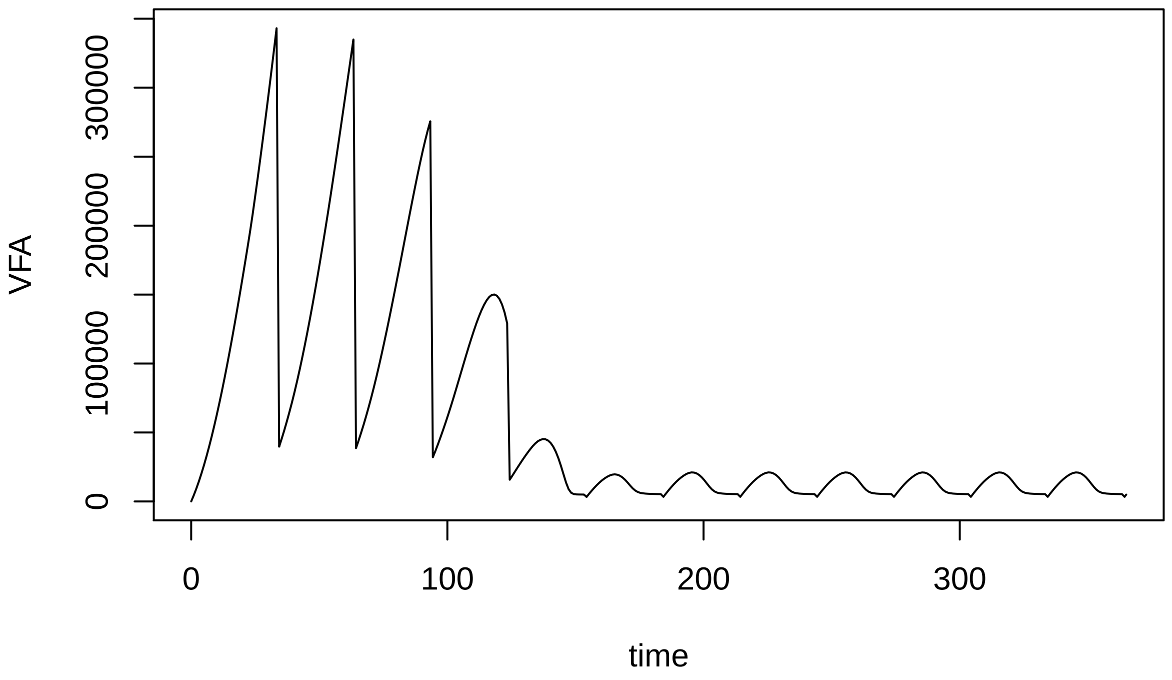
Because of a default temperature of 23 (NTS: why so high???) methanogen `m3` dominates under default conditions. Biomass concentrations (g per kg of slurry) may be more informative.

```r
plot(m3_conc ~ time, data = out1, type = 'l')
```

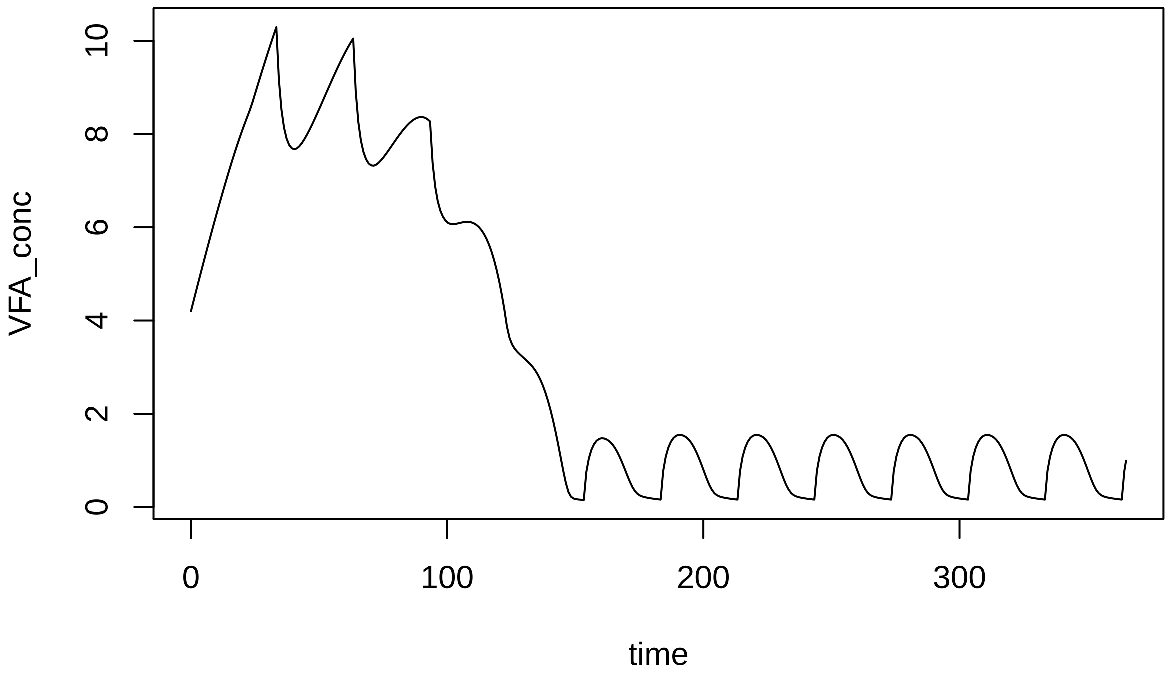


Dynamics in production of CH_4 are often related to VFA accumulation, and VFA mass (g) and concentration (g/kg) can be extracted.

```
plot(VFA ~ time, data = out1, type = 'l')
```



```
plot(VFA_conc ~ time, data = out1, type = 'l')
```



For more information on the many output variables returned by `abm()`, see the section on that topic below.

4. Setting parameter values

Although the ABM model is relatively simple, explicitly simulating the activity of multiple microbial groups means there are many parameters. The complete list and definitions can be seen in the help file, accessible with the following command.

```
?abm
```

Alternatively, use `args()` just to see the arguments and default values.

```
args(abm)
```

```
## function (days = 365, delta_t = 1, mng_pars = list(slurry_prod_rate = 1000,
##     slurry_mass = 0, max_slurry_mass = 33333, resid_frac = 0.1,
##     area = 11, temp_C = 23), man_pars = list(conc_fresh = list(S2 = 0,
##     SO4 = 0.2, TAN = 1, VFA = 4.2, Sp = 65, COD = 160), pH = 7),
##     grp_pars = list(yield = c(default = 0.04, sr1 = 0.065), xa_fresh = c(default = 0.001,
##         sr1 = 0.001), xa_init = c(m1 = 0.01, m2 = 0.01, m3 = 0.01,
##         p1 = 0.01, p2 = 0.01, sr1 = 0.01), decay_rate = c(m1 = 0.02,
##         m2 = 0.02, m3 = 0.02, p1 = 0.02, p2 = 0.02, sr1 = 0.02),
##         ks_coefficient = c(m1 = 0.5, m2 = 1.5, m3 = 1, p1 = 1,
##             p2 = 1, sr1 = 0.4), resid_enrich = c(m1 = 0, m2 = 0,
##             m3 = 0, p1 = 0, p2 = 0, sr1 = 0), qhat_opt = c(m1 = 8,
##             m2 = 13.33, m3 = 5.75, p1 = 2.77, p2 = 0.72, sr1 = 8.3),
##         T_opt = c(m1 = 313, m2 = 313, m3 = 303, p1 = 293, p2 = 283,
##             sr1 = 313), T_min = c(m1 = 295.31, m2 = 295.31, m3 = 285.31,
##             p1 = 275.31, p2 = 265.31, sr1 = 273), T_max = c(m1 = 320.67,
##             m2 = 320.67, m3 = 310.67, p1 = 300.67, p2 = 290.67,
##             sr1 = 320.67), ki_NH3_min = c(m1 = 0.01, m2 = 0.015,
##             m3 = 0.015, p1 = 0.015, p2 = 0.015, sr1 = 0.015),
##         ki_NH3_max = c(m1 = 0.1, m2 = 0.131, m3 = 0.131, p1 = 0.131,
##             p2 = 0.131, sr1 = 0.131), ki_NH4_min = c(m1 = 1.7,
##             m2 = 2.714, m3 = 2.714, p1 = 2.714, p2 = 2.714, sr1 = 2.714),
##         ki_NH4_max = c(m1 = 3.1, m2 = 4.764, m3 = 4.764, p1 = 4.764,
##             p2 = 4.764, sr1 = 4.764), pH_upr = c(m1 = 8, m2 = 8,
##             m3 = 8, p1 = 8, p2 = 8, sr1 = 8), pH_lwr = c(m1 = 6.5,
##             m2 = 6, m3 = 6.5, p1 = 6.5, p2 = 6.5, sr1 = 6)),
##     mic_pars = list(ks_SO4 = 0.0067, ki_H2S_meth = 0.23, ki_H2S_sr = 0.25,
##         alpha_opt = 0.015, alpha_T_opt = 313, alpha_T_min = 273,
##         alpha_T_max = 320.67), chem_pars = list(COD_conv = c(CH4 = 0.2507,
##         S = 0.5015, VS = 0.69, CO2_anaer = 0.57, CO2_aer = 1.3,
##         CO2_sr = 1.3), kl = c(H2S = 0.032, oxygen = 0.415)),
##     add_pars = NULL, startup = -Inf, starting = NULL, approx_method_temp = "linear",
##     approx_method_pH = "linear", approx_method_SO4 = "linear",
##     par_key = "\\.", value = "ts", warn = TRUE)
## NULL
```

Parameters are grouped to make changes easier (and to prevent input mistakes) and to limit the number of parameter names that are needed. The `mng_pars` argument contains parameters related to management; `man_pars` describes the incoming manure or feed; `grp_pars`, the most extensive argument, is used to define the microbial groups; `mic_pars` contains other microbial parameters that do not vary among groups; and `chem_pars` sets some chemical/physical parameters. But there are also some built-in shortcuts to make small tweaks simple. In particular, the `add_pars` argument makes life easy.

As an example, the composition of the fresh slurry (influent, or feed) is set with the `man_pars` argument, which is a list of solute concentrations and pH. By default:

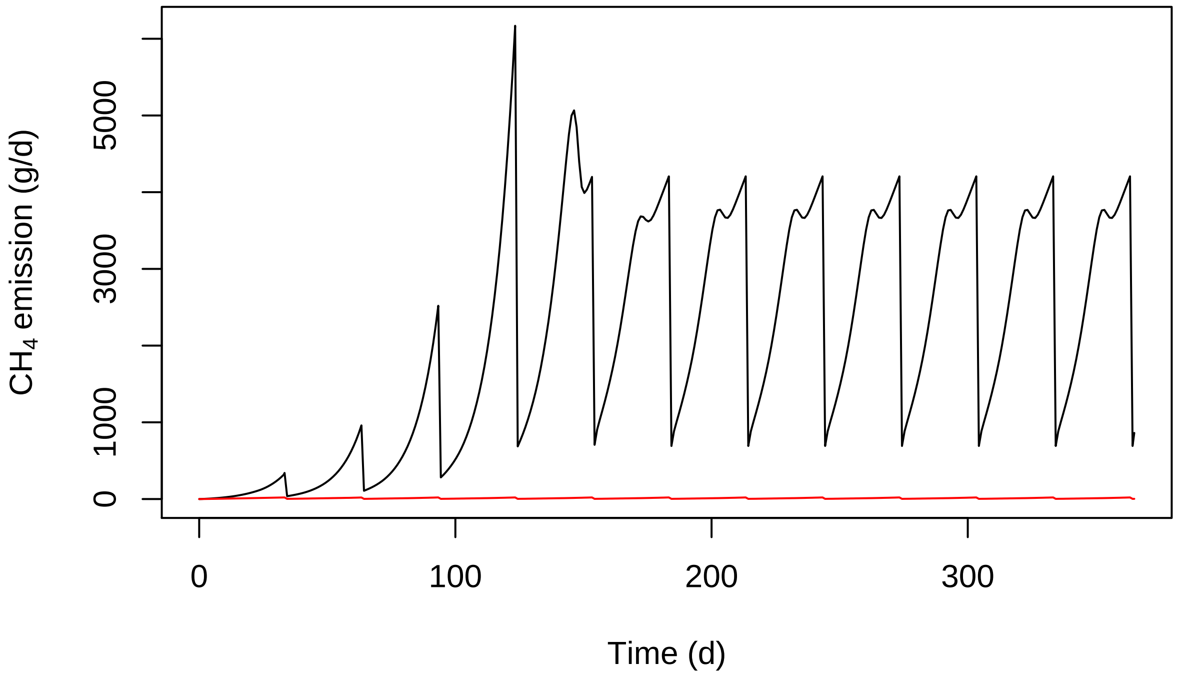
```
man_pars = list(conc_fresh = list(S2 = 0.0, SO4 = 0.2, TAN = 1.0,
                                  VFA = 4.0, Sp = 65, COD = 170),
                pH = 7), ...
```

To simulate a lower pH then, the following call could be used:

```r
out2 <- abm(365, 1, man_pars = list(conc_fresh = list(S2 = 0.0, SO4 = 0.2, TAN = 1.0,
                                                       VFA = 4.2, Sp = 65, COD = 160),
                                   pH = 6))
```

Below CH_4 emission rate is compared to the default predictions.

```r
plot(CH4_emis_rate ~ time, data = out1, type = 'l', xlab = 'Time (d)',
     ylab = expression('CH'[4]~'emission (g/d)'))
lines(CH4_emis_rate ~ time, data = out2, type = 'l', col = 'red')
```



Alternatively, the special `add_pars` argument can be used to specify just those parameters (or individual parameter elements) that will be changed from their defaults.

```r
out2b <- abm(365, 1, add_pars = list(pH = 6))
```

These two approaches provide identical results:

```r
all.equal(out2, out2b)
```

```
## [1] TRUE
```

Note that the `man_pars` name is not needed for the `add_pars` option.

Many arguments for the `abm()` function are named lists or vectors. These arguments–or even one element within them–can still be specified using `add_pars`. For example, to change only the VFA value for `conc_fresh` the following call provides a shortcut compared to specifying all elements within the `conc_fresh` vector (as in the `out2` example above).

```r
out3 <- abm(365, 1, add_pars = list(pH = 6, conc_fresh.VFA = 10))
```

This shortcut is referred to as the "par.element" approach in the documentation, and the `.` is a special character used to separate parameter (here, `conc_fresh`) and element (here, `VFA`) names. (If desired, a different character can be set with the `par_key` argument.)

Of course, specifying all elements is always an option,

```
out3b <- abm(365, 1, add_pars = list(pH = 6, conc_fresh = list(S2 = 0.0, SO4 = 0.2,
                                                              TAN = 1.0, VFA = 10,
                                                              Sp = 65, COD = 160)))
```

as is specifying a complete argument of parameters (as in `out2` above).

Setting arguments is explored further in the section on defining microbial groups below (Section X).

5. Output options

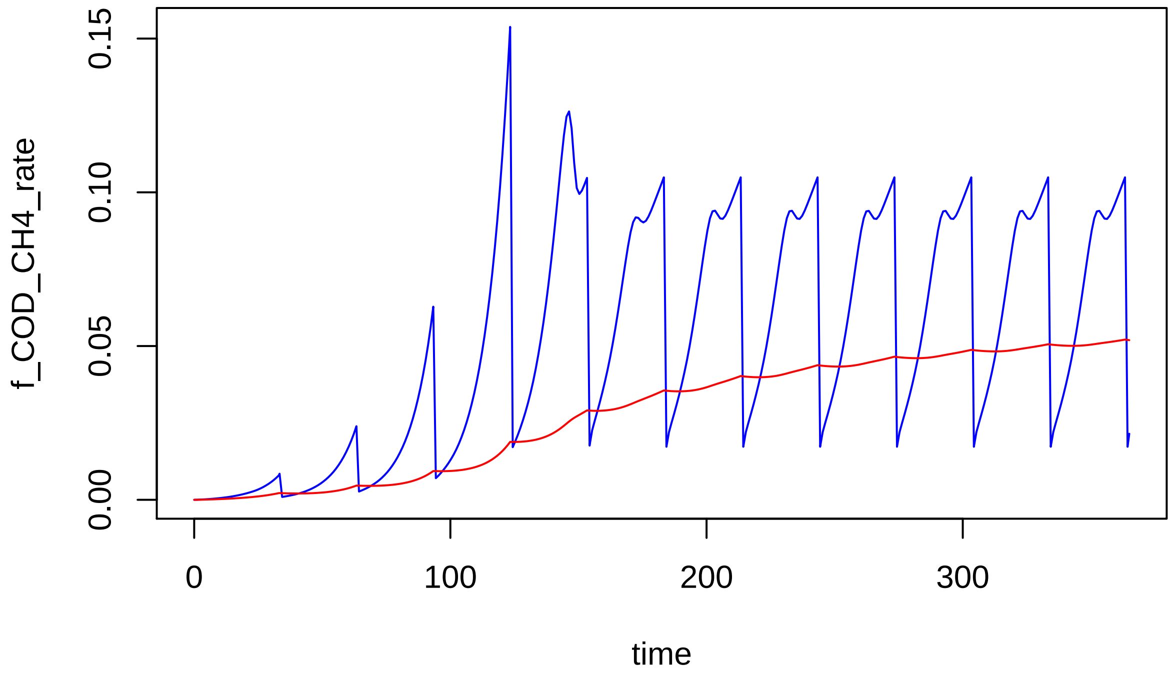
By default, the `abm()` function returns a data frame with cumulative CH_4 emission and other state variables, normalized in a variety of ways. In total there are more than 300 columns—the first 20 are shown below.

```
out1 <- abm(365, 1)
out1[365, 1:20]
```

```
##         time       m1       m2       m3       p1       p2     sr1 slurry_mass
## 365 363.3297 24.33115 24.46087 11224.09 88.67118 23.86923 638.915       33333
##          Sp      VFA sulfate  sulfide CH4_emis_cum CO2_emis_cum COD_conv_cum
## 365 1922079 5301.607 76.8476 3610.787     762829.2      1940205      3201113
##     COD_conv_cum_meth COD_conv_cum_respir COD_conv_cum_sr       NH4        NH3
## 365           3042797            14872.19        143443.9 0.9950865 0.00491348
```

Microbial biomass values (g COD) are present in the columns that directly follow time (d). Emission of CH_4 and CO_2 are included as cumulative values (g), rates (g/d), and both types are also normalized by loading of COD, degradable COD (`dCOD`), and VS (based on either instantaneous rates or cumulative values). The fraction of loaded COD converted through methanogenesis, respiration, and sulfate reduction is also given–these variables start with `f_`. For example, fractional conversion of COD to CH_4 based on instantaneous rates and cumulative values are shown in the plot below.

```
plot(f_COD_CH4_rate ~ time, data = out1, type = 'l', col = 'blue')
lines(f_COD_CH4_cum ~ time, data = out1, col = 'red')
```



Overall results can be extracted by changing the `value` argument to `sum` (for summary).

```r
out1s <- abm(365, 1, value = 'sum')
out1s
```

```
##        COD_load       dCOD_load      ndCOD_load         VS_load     CH4_emis_cum
##    5.872000e+07    2.539860e+07    3.332140e+07    4.051680e+07    7.640988e+05
##   CH4_emis_rate    CH4_emis_COD   CH4_emis_dCOD     CH4_emis_VS    CO2_emis_cum
##    2.093421e+03    1.301258e-02    3.008429e-02    1.885881e-02    1.943565e+06
##   CO2_emis_rate    CO2_emis_COD   CO2_emis_dCOD     CO2_emis_VS   COD_conv_meth
##    5.324835e+03    3.309886e-02    7.652251e-02    4.796936e-02    3.047861e+06
## COD_conv_respir     COD_conv_sr       f_COD_CH4    f_COD_respir        f_COD_sr
##    1.494056e+04    1.437395e+05    5.190499e-02    2.544374e-04    2.447879e-03
```

And an arbitrary startup period can be excluded from these summary results using the `startup` argument. For example, the first 100 days are excluded in the example below.

```r
out1s <- abm(365, 1, value = 'sum', startup = 100)
out1s
```

```
##        COD_load       dCOD_load      ndCOD_load         VS_load     CH4_emis_cum
##    4.256000e+07    1.840880e+07    2.415120e+07    2.936640e+07    7.256779e+05
##   CH4_emis_rate    CH4_emis_COD   CH4_emis_dCOD     CH4_emis_VS    CO2_emis_cum
##    2.741846e+03    1.705070e-02    3.942017e-02    2.471116e-02    1.799578e+06
##   CO2_emis_rate    CO2_emis_COD   CO2_emis_dCOD     CO2_emis_VS   COD_conv_meth
##    6.799390e+03    4.228332e-02    9.775643e-02    6.128018e-02    2.894607e+06
## COD_conv_respir     COD_conv_sr       f_COD_CH4    f_COD_respir        f_COD_sr
##    1.083365e+04    1.042836e+05    6.801238e-02    2.545501e-04    2.450273e-03
```

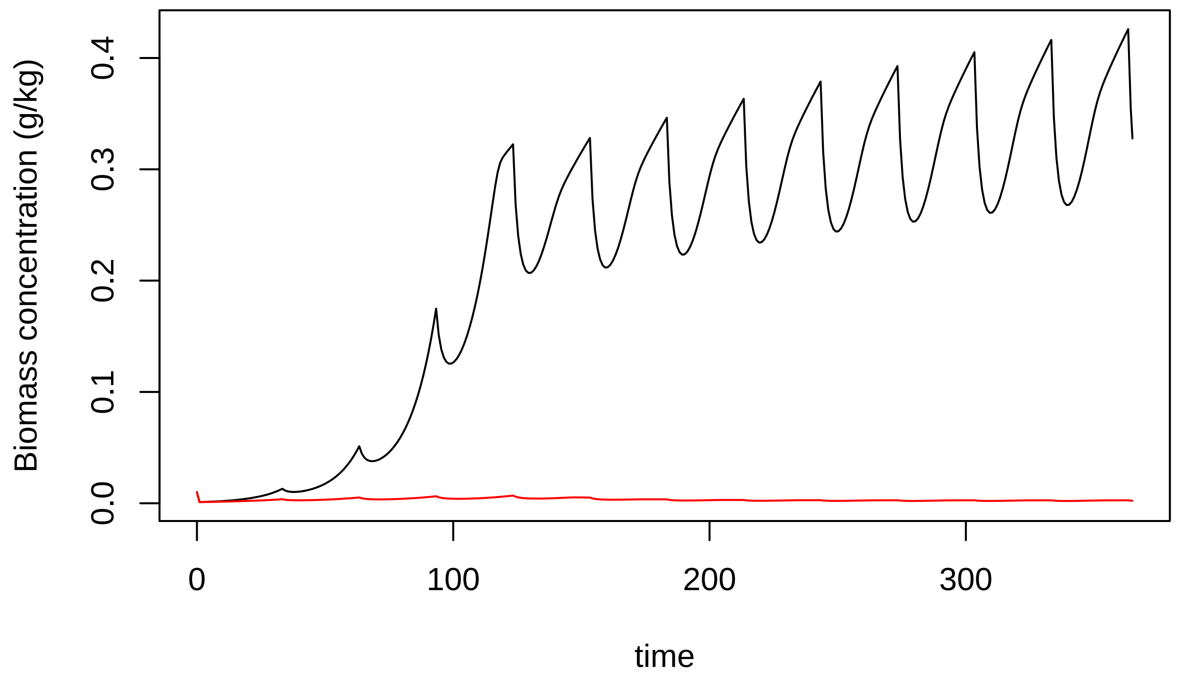
Alternatively, set the `value` argument to `'all'` for time series data and the summary.

6. Defining microbial groups

By default, the ABM model includes six microbial groups: five methanogens and one sulfate reducer. Each microbial group is characterized by 14 parameters that describe the rate of metabolism, biomass yield, decay rate, and the response to temperature, pH, and ammonia. Additionally, values are needed for biomass concentrations in fresh slurry and the storage. Lastly, an enrichment factor parameter is specified for each group. Unlike VFA consumption, the rate of the combined hydrolysis and fermentation step is controlled by a simple temperature-dependent first-order rate constant. Aerobic respiration is controlled by the mass transfer rate of O_2 to the slurry surface. For these two processes then, there is no (explicit) associated microbial group.

A central feature of the ABM model is the ability to specify any number of methanogenic groups. To define a custom set, a single (albeit complex) argument `grp_pars` needs to be set (see Section X). This task is straightforward if tedious, and described after a description of tweaking parameters. A more common need is to tweak default parameters. Although this can also be done using `grp_pars`, it is more efficient to use `add_pars`. For example, to increase `qhat_optim` of group `p1` to 3 g/g-d (g substrate COD per g biomass COD per day) and the yield to 0.06 g/g, the following call could be used:

```r
out4 <- abm(365, 1, add_pars = list(qhat_opt.p1 = 3, yield.p1 = 0.06))
plot(p1_conc ~ time, data = out4, type = 'l', ylab = 'Biomass concentration (g/kg)')
lines(p1_conc ~ time, data = out1, type = 'l', col = 'red')
```



This change from the default values (2.77 and 0.04) has a drastic effect, which perhaps should not be too surprising because the yield change alone represents a 50% improvement in fitness.

To define completely new microbial groups, the example of the default argument in the help file can be followed. The special keywords `all` and `default` make the task easier. All groups must be named individually in the `qhat_opt` parameter, which effectively identifies the groups. For other parameters though, use `all` if all groups have the same value (as in `xa_init` below). Or, use `default` if some groups have the same value (as in `xa_fresh` below).

```
grp_pars <- list(yield = c(default = 0.04, sr1 = 0.065),
                 xa_fresh = c(default = 0.001, sr1 = 0.001),
                 xa_init = c(all = 0.01),
                 decay_rate = c(all = 0.02),
                 ks_coefficient = c(m1 = 0.5, m2 = 1.5, m3 = 1.0, p1 = 1.0, p2 = 1.0, sr1 = 0.4),
                 resid_enrich = c(all = 0),
                 qhat_opt = c(m1 = 8, m2 = 13.33, m3 = 5.75, p1 = 2.77, p2 = 0.72, sr1 = 8.3),
                 T_opt = c(m3 = 303, p1 = 293, p2 = 283, default = 313),
                 T_min = c(m1 = 295.31, m2 = 295.31, m3 = 285.31, p1 = 275.31, p2 = 265.31, sr1 = 273),
                 T_max = c(m1 = 320.67, m2 = 320.67, m3 = 310.67, p1 = 300.67, p2 = 290.67, sr1 = 320.6
                 ki_NH3_min = c(m1 = 0.01, m2 = 0.015, m3 = 0.015, p1 = 0.015, p2 = 0.015, sr1 = 0.015)
                 ki_NH3_max = c(m1 = 0.10, m2 = 0.131, m3 = 0.131, p1 = 0.131, p2 = 0.131, sr1 = 0.131)
                 ki_NH4_min = c(m1 = 1.70, m2 = 2.714, m3 = 2.714, p1 = 2.714, p2 = 2.714, sr1 = 2.714)
                 ki_NH4_max = c(m1 = 3.10, m2 = 4.764, m3 = 4.764, p1 = 4.764, p2 = 4.764, sr1 = 4.764)
                 pH_upr = c(m1 = 8.0, m2 = 8.0, m3 = 8.0, p1 = 8.0, p2 = 8.0, sr1 = 8.0),
                 pH_lwr = c(m1 = 6.5, m2 = 6.0, m3 = 6.5, p1 = 6.5, p2 = 6.5, sr1 = 6.0))

out5 <- abm(365, 1, grp_pars = grp_pars)
```

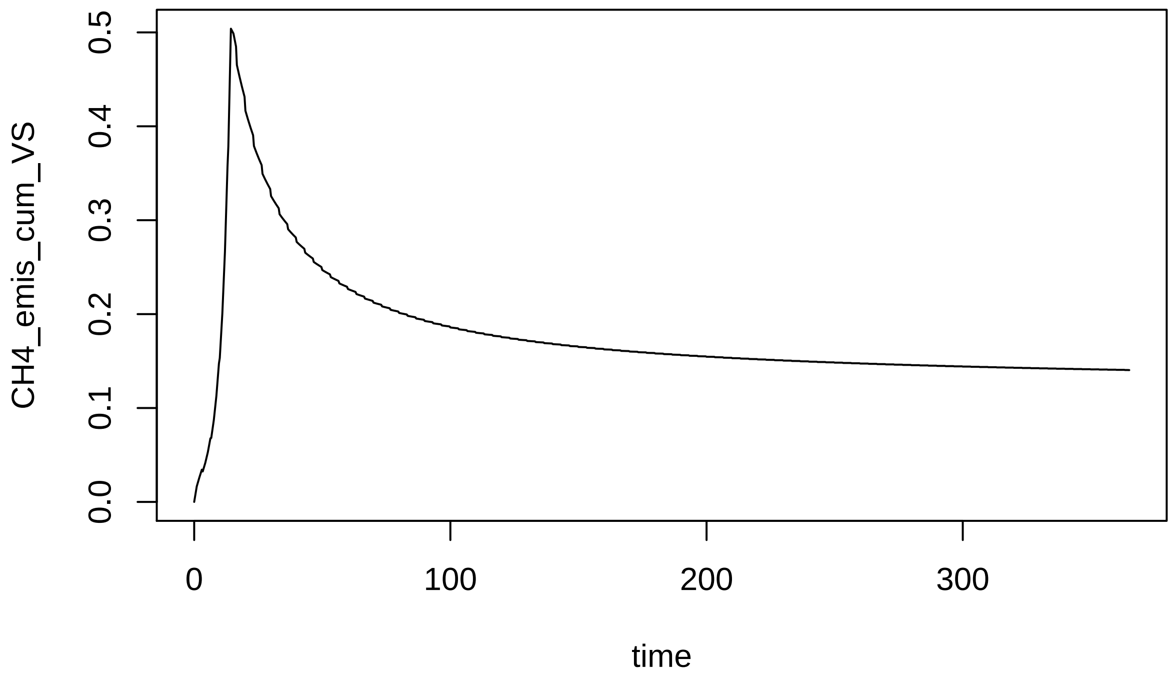
7. Simulating reactors

The ABM model inherently describes a reactor with continuous feeding and intermittent wasting. To approximate a continuous reactor (which is not actually "continuous" in practice but typically has intermittent feeding and wasting–but this is a seprate discussion) the `resid_frac` argument can be set to a high value, e.g. `0.95`. This provides frequent wasting of a small quantity. The following example simulates the startup of a mesophilic completely mixed anaerobic digester fed cattle manure (based on defaults).

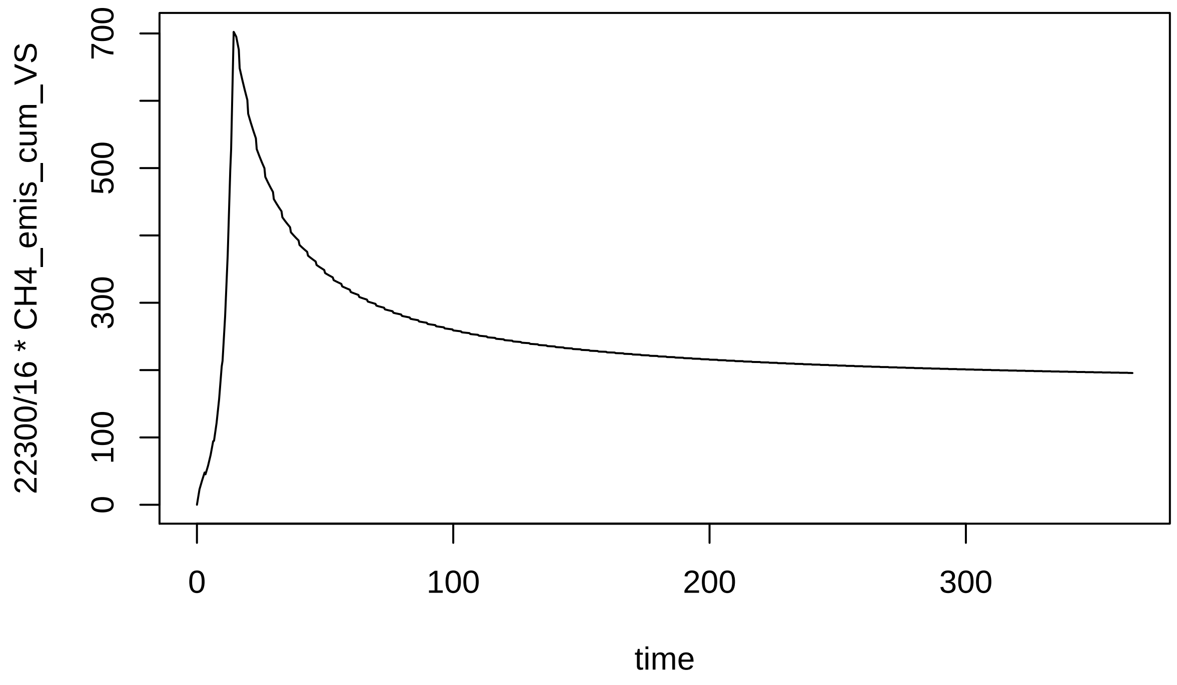
```r
out5 <- abm(365, 1, add_pars = list(temp_C = 35, resid_frac = 0.95, alpha_opt = 0.2,
                                    slurry_mass = 0.95 * 33333, slurry_prod_rate = 500))
```

Due to the structure of the code (the ODE solver is called separately for each filling interval), a drawback of this high `resid_frac` approach is a long evaluation time.

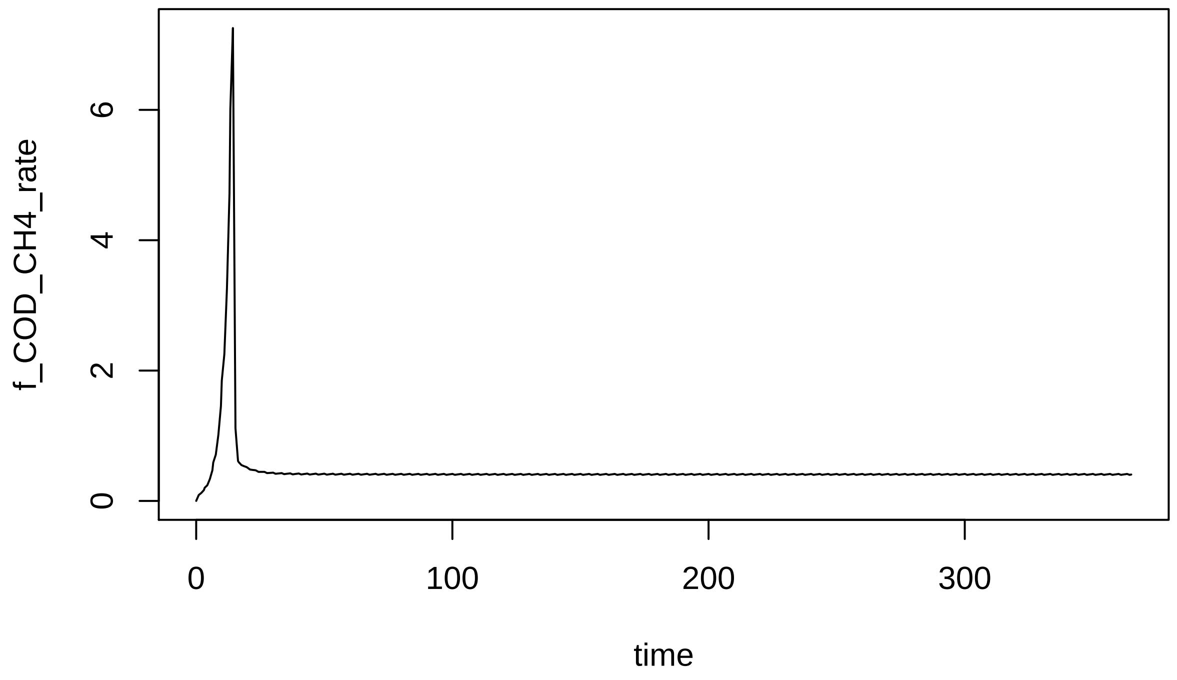
```r
plot(CH4_emis_cum_VS ~ time, data = out5, type = 'l')
```



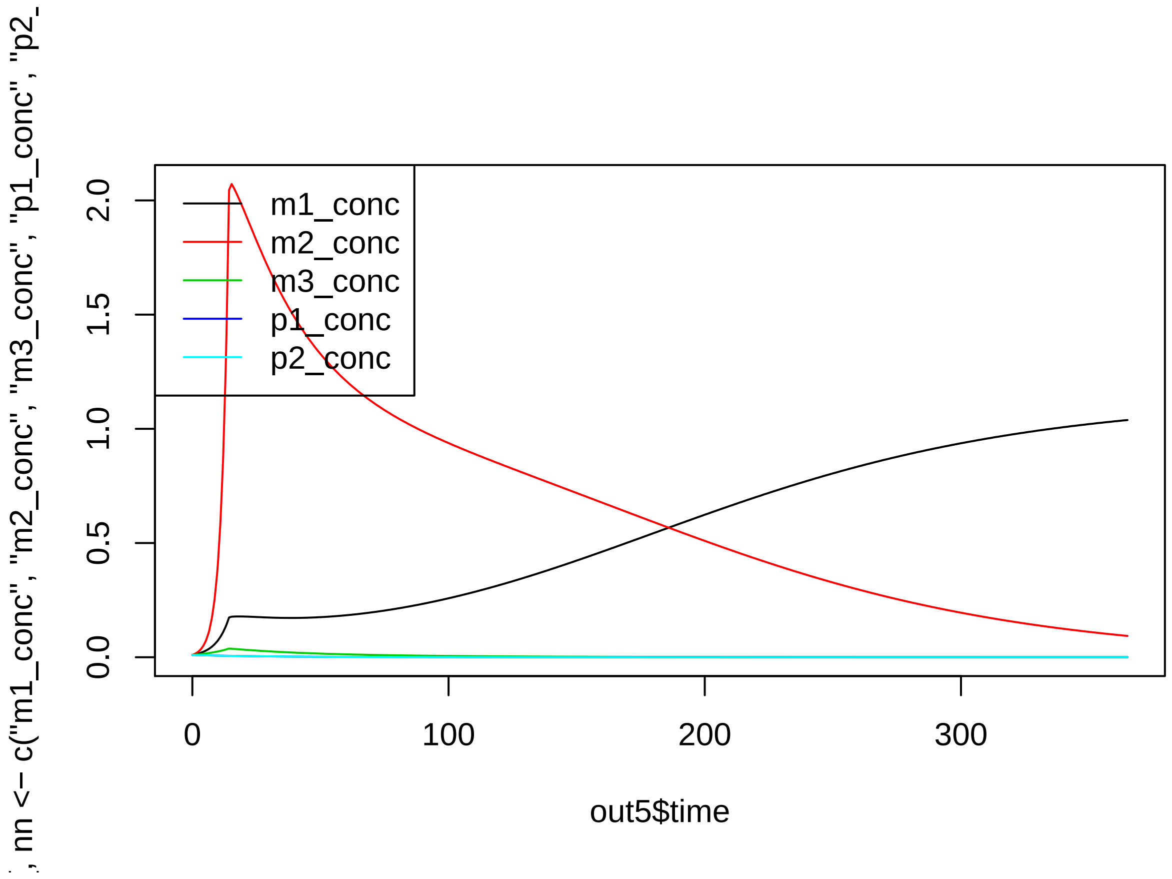
```r
plot(22300 / 16 * CH4_emis_cum_VS ~ time, data = out5, type = 'l')
```



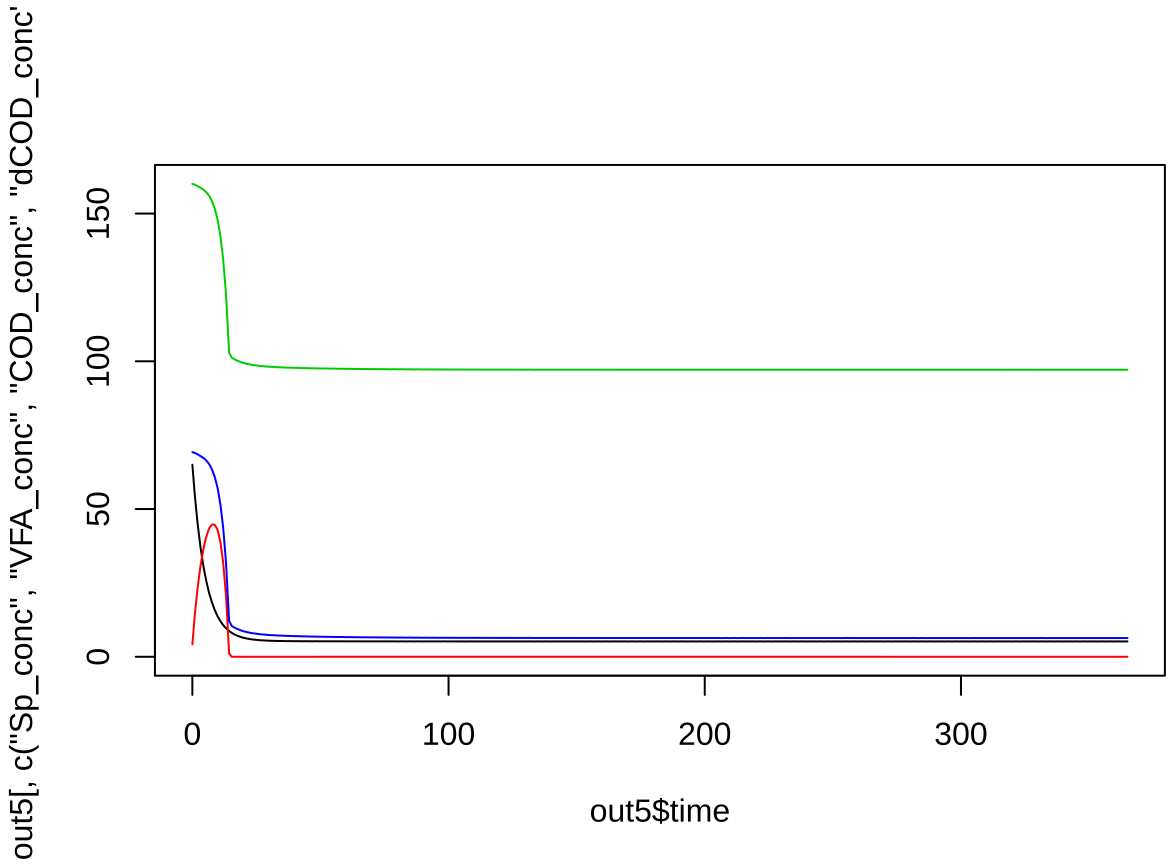
```r
plot(f_COD_CH4_rate ~ time, data = out5, type = 'l')
```



```r
matplot(out5$time, out5[, nn <- c('m1_conc', 'm2_conc', 'm3_conc', 'p1_conc', 'p2_conc')],
        type = 'l', lty = 1)
legend('topleft', nn, col = 1:5, lty = 1)
```



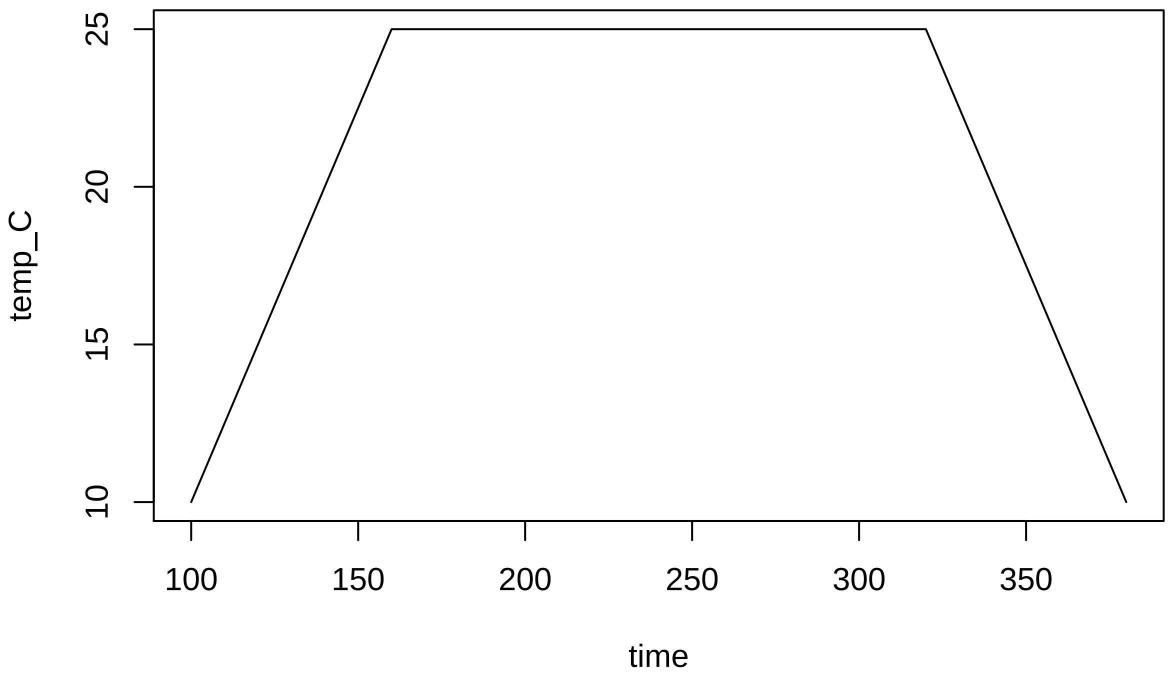
```r
matplot(out5$time, out5[, c('Sp_conc', 'VFA_conc', 'COD_conc', 'dCOD_conc')],
        type = 'l', lty = 1)
```



8. Variable temperature

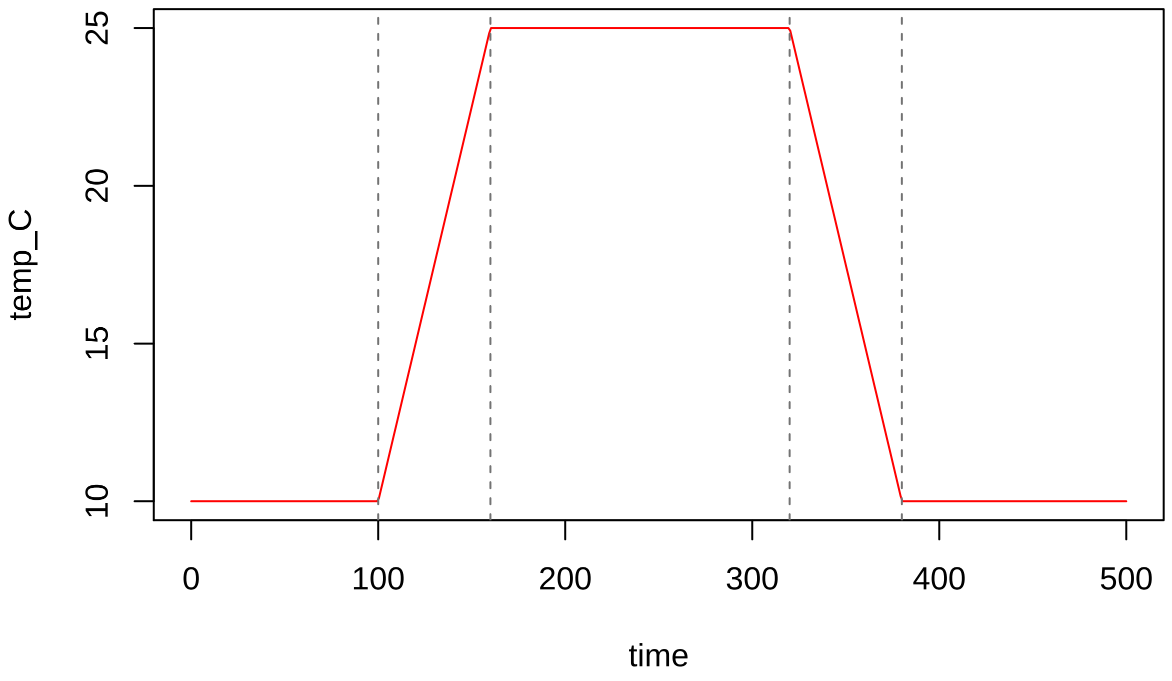
Predicting short- and long-term responses to temperature change was a central objective of the ABM model. Variable temperature is entered in a data frame with two columns. For example, gradual warming from 10°C to 25°C, a hold, and then a gradual cooling back to 10°C can be specified as shown in the `temp_dat` data frame constructed below.

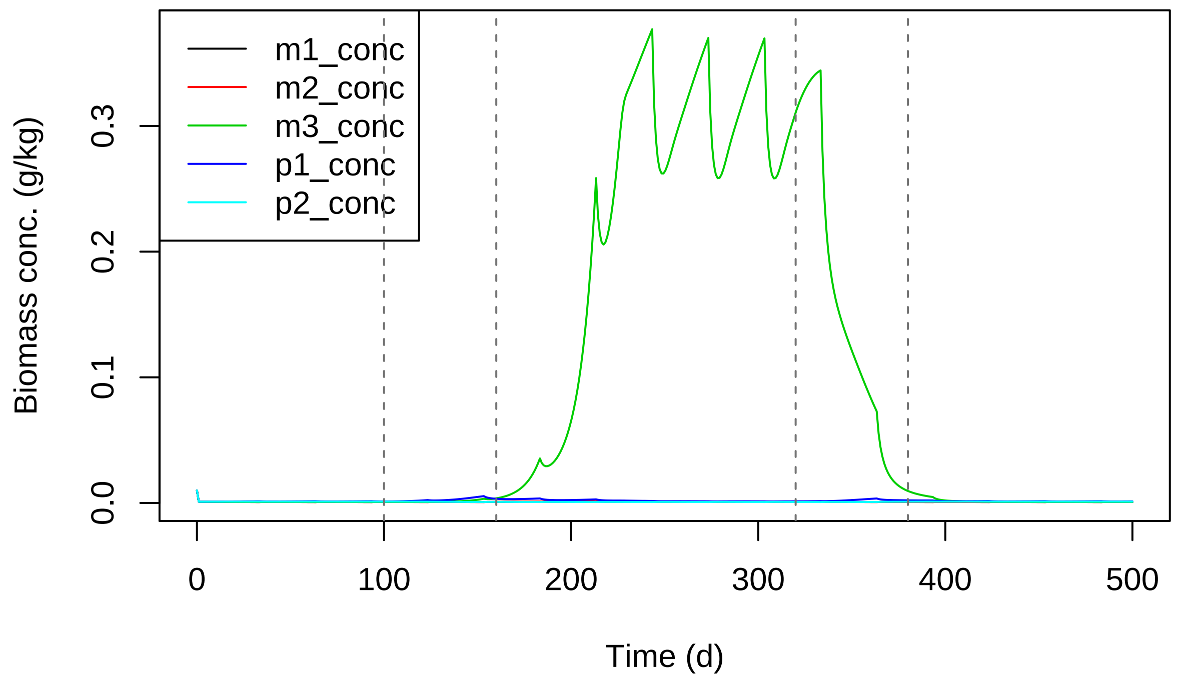
```
temp_dat <- data.frame(time = 100 + c(0, 60, 220, 280),
                       temp_C =     c(10, 25,  25,   10))
plot(temp_C ~ time, data = temp_dat, type = 'l')
```

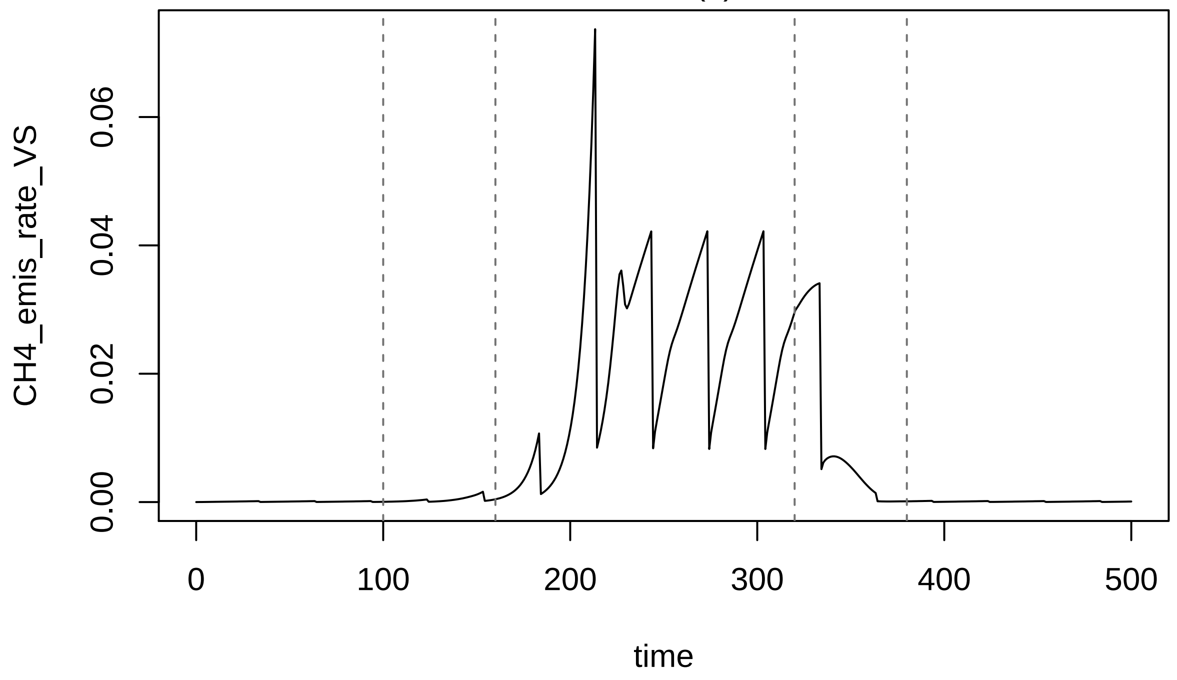


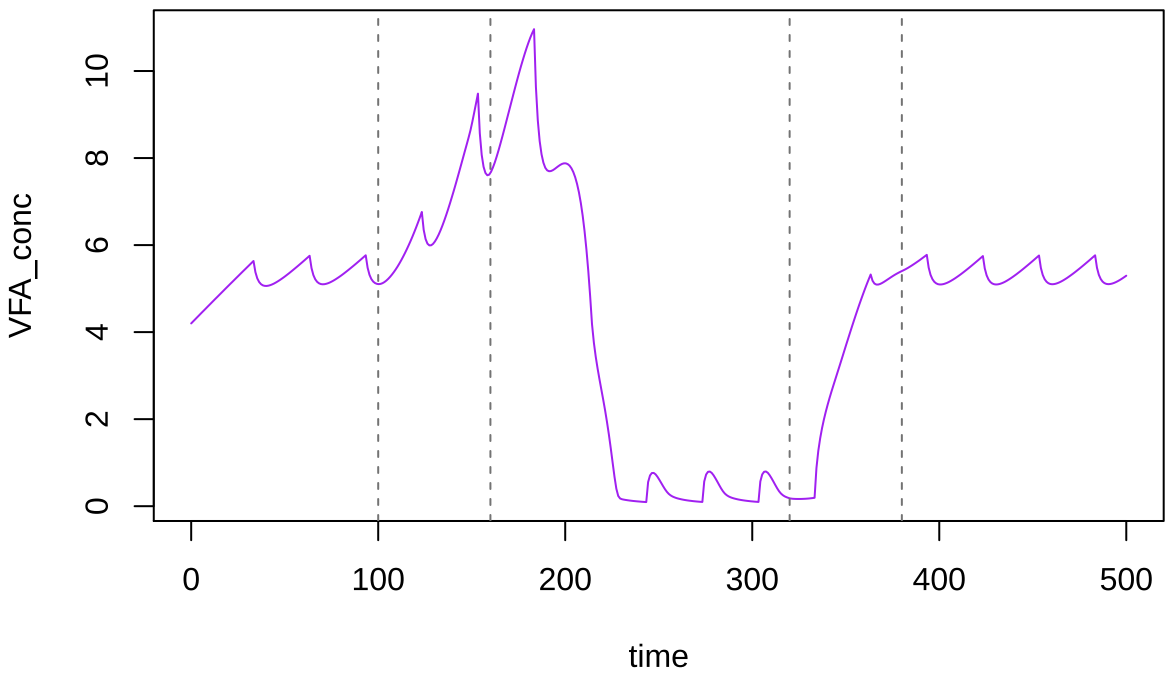
The model can either interpolate (the default) or use constant temperatures between change points. The temperature data can be supplied using the `mng_pars` argument or, more simply, with `add_pars`.

```
out6 <- abm(500, 1, add_pars = list(temp_C = temp_dat))
```




m1_conc
m2_conc
m3_conc
p1_conc
p2_conc
Biomass conc. (g/kg)
0.0
0.1
0.2
0.3
0
100
200
300
400
500
Time (d)


CH4_emis_rate_VS
0.00
0.02
0.04
0.06
0
100
200
300
400
500
time

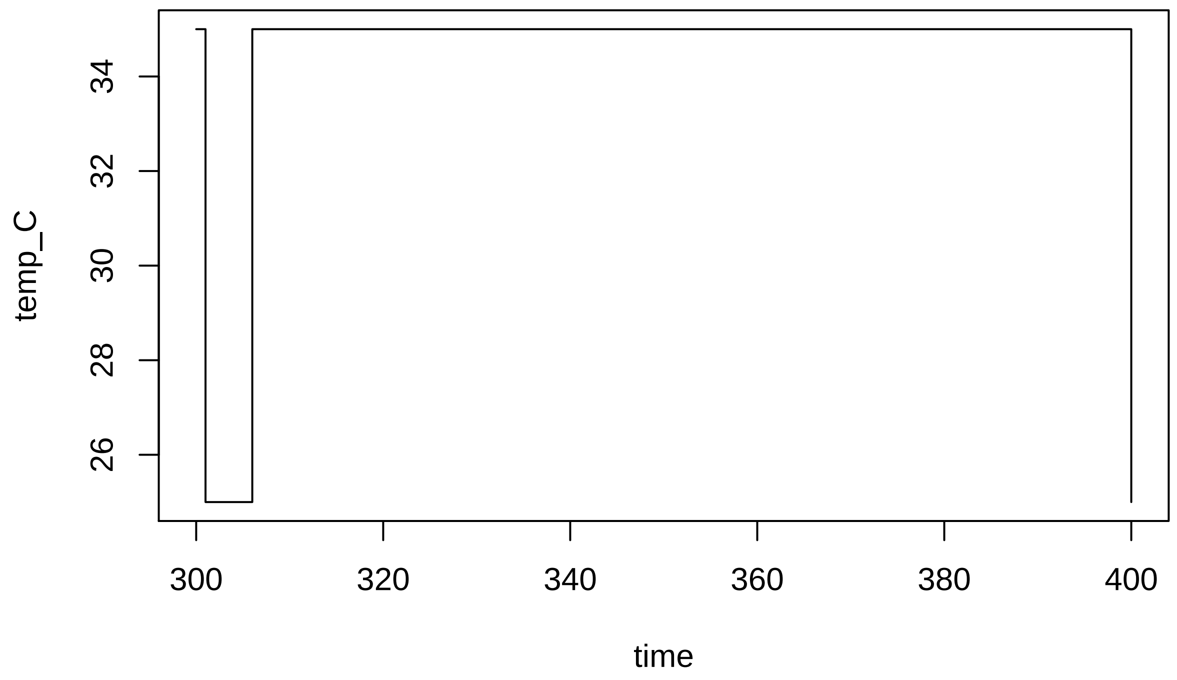


NTS: Really no development of a psychrophilic population? NTS: What does it take?

Temperature data could be much higher resolution, e.g., daily values.

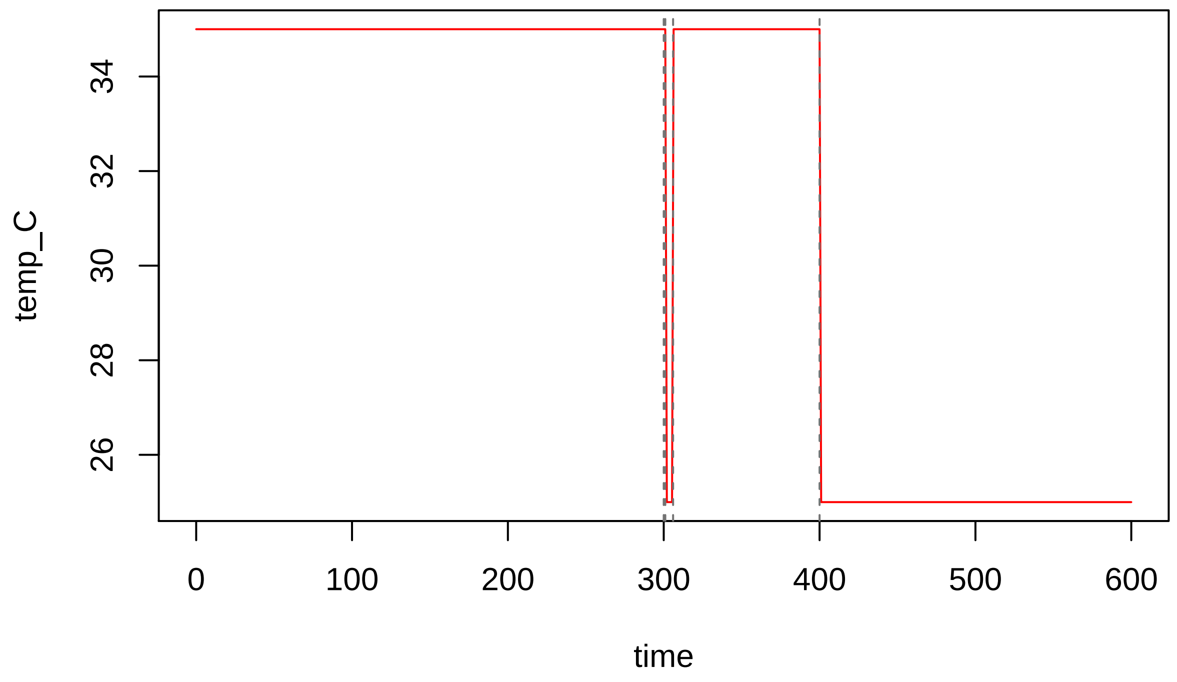
For anaerobic digesters, the effect of both short- and long-term changes are of interest. In a controlled environment, temperature change is not always gradual, but can be (deliberately) rapid. The `approx_method_temp` argument can be used for this type of pattern, instead of the linear interpolation shown above (which is the default). The following data frame can be used to simulate a reactor initially running at 35°C suddenly reduced to 25°C for 5 days, followed by stabilization and finally a much longer temperature change.

```
temp_dat <- data.frame(time = 300 + c(0,  1,  6, 100),
                       temp_C =    c(35, 25, 35,  25))
plot(temp_C ~ time, data = temp_dat, type = 's')
```

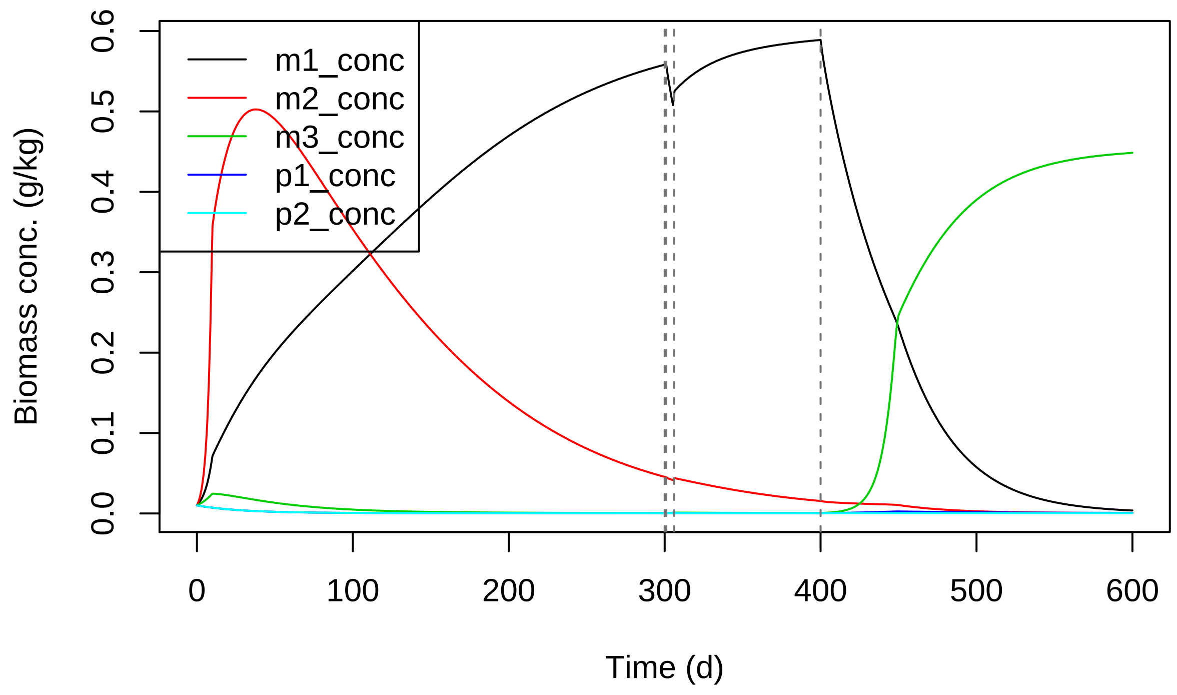


```
out7 <- abm(600, 1, add_pars = list(temp_C = temp_dat, resid_frac = 0.95,
                                        slurry_mass = 0.95 * 33333, slurry_prod_rate = 500),
            approx_method_temp = 'constant')
```

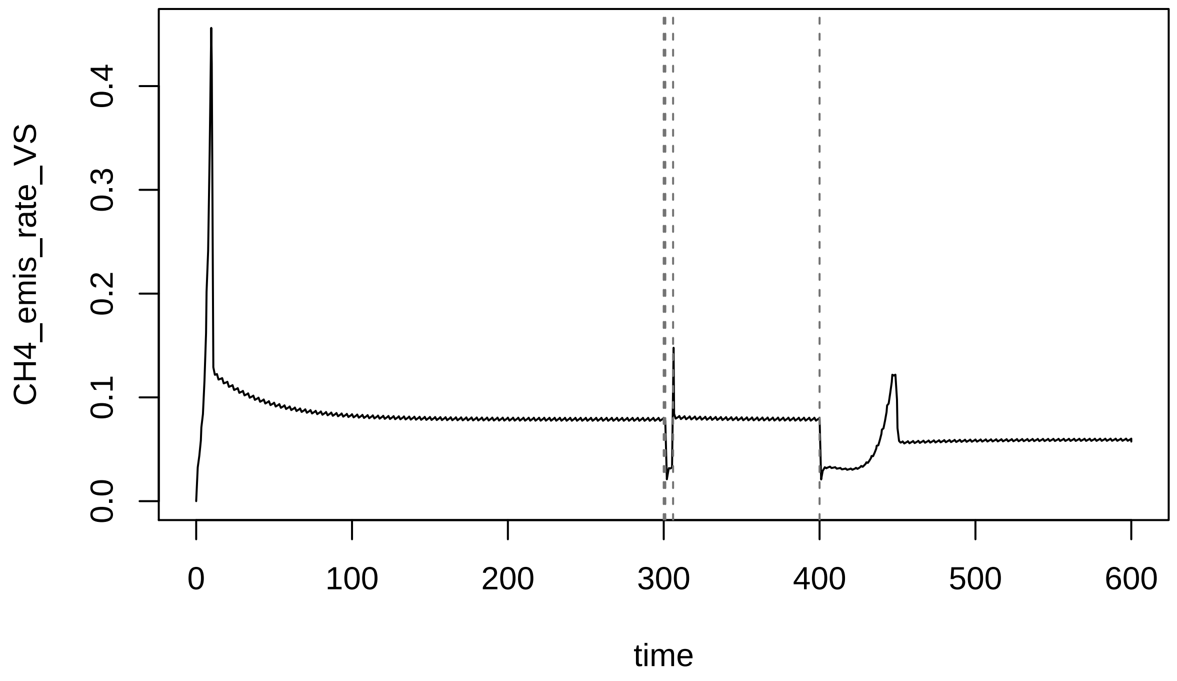
```
plot(temp_C ~ time, type = 'l', col = 'red', data = out7)
abline(v = temp_dat$time, lty = 2, col = 'gray45')
```



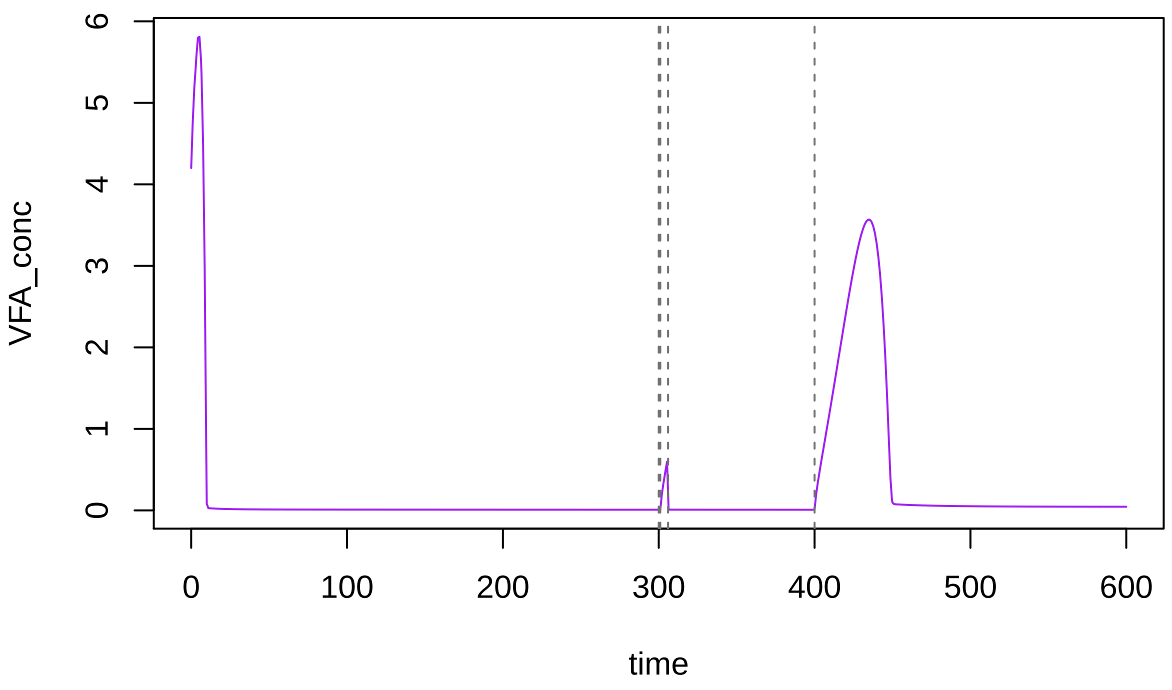
```
matplot(out7$time, out7[, nn <- c('m1_conc', 'm2_conc', 'm3_conc', 'p1_conc', 'p2_conc')],
        type = 'l', lty = 1, xlab = 'Time (d)', ylab = 'Biomass conc. (g/kg)')
legend('topleft', nn, col = 1:5, lty = 1)
abline(v = temp_dat$time, lty = 2, col = 'gray45')
```



```r
plot(CH4_emis_rate_VS ~ time, data = out7, type = 'l')
abline(v = temp_dat$time, lty = 2, col = 'gray45')
```



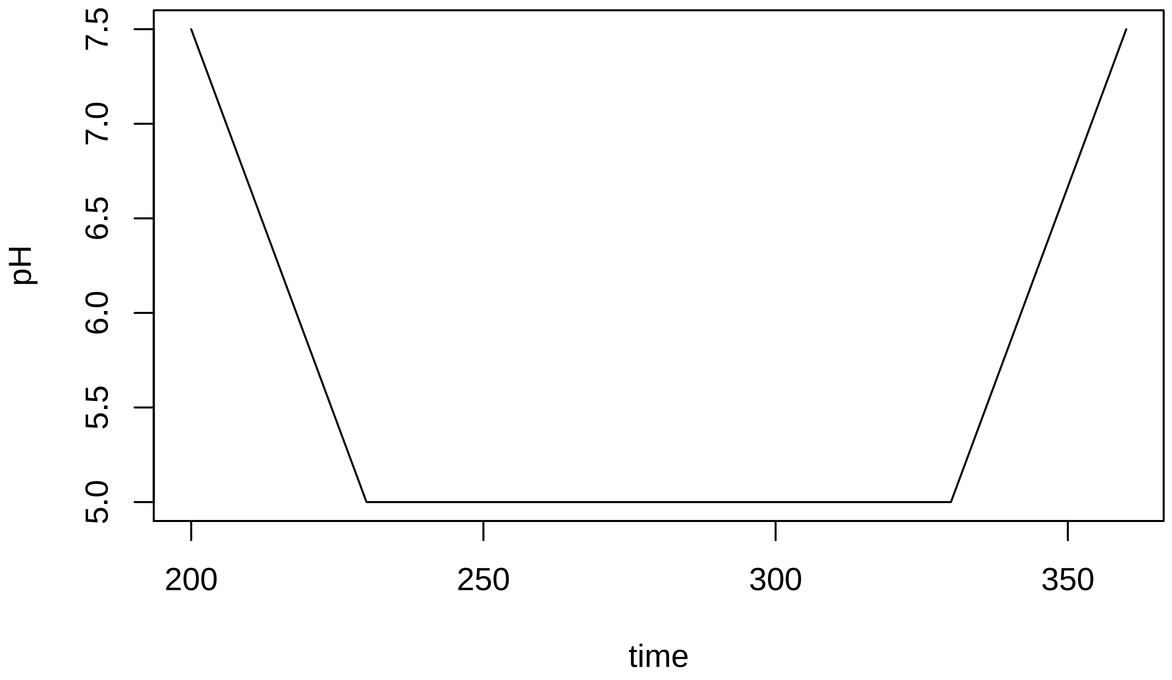
```r
plot(VFA_conc ~ time, type = 'l', col = 'purple', data = out7)
abline(v = temp_dat$time, lty = 2, col = 'gray45')
```



8. Acidification

Acidification of slurry with sulfuric acid (H_2SO_4) is an effective approach for reducing CH_4 emission. In the ABM model, there are a few options for how acidification can be specified. With data on slurry pH vs. time, the `pH` parameter can be used. As with temperature, both instant changes and linear interpolation can be used. The following data might come from a channel where acidification took effect over 30 days, was used for 100, and then stopped.

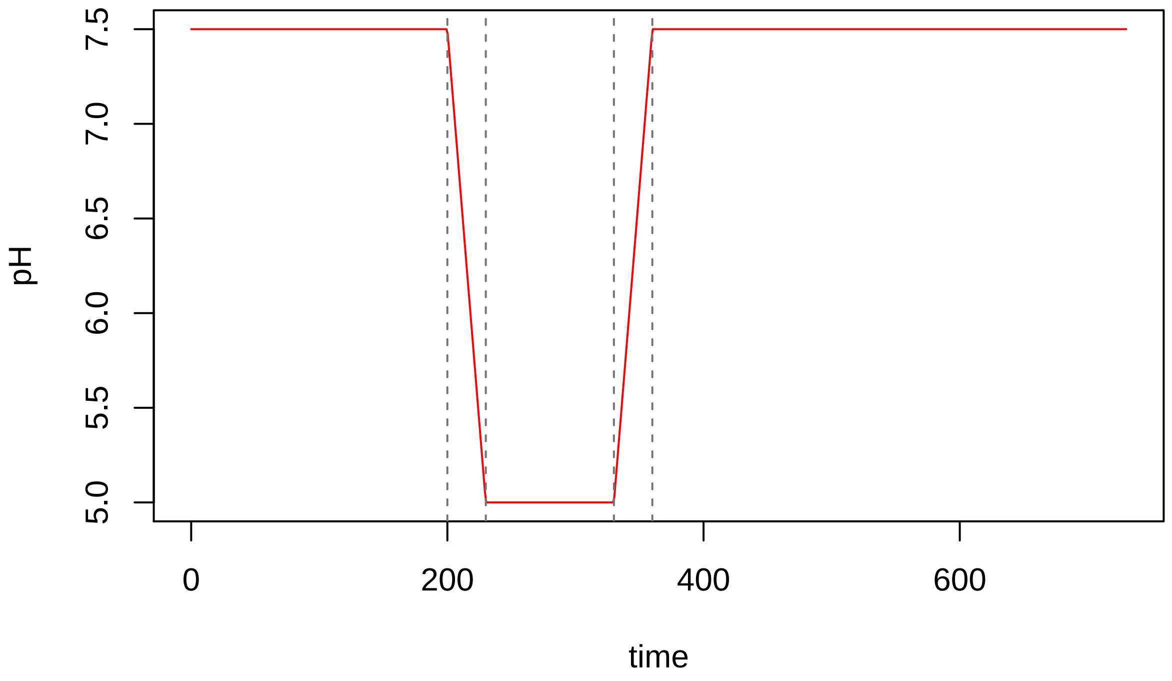
```r
pH_dat <- data.frame(time = c(200, 230, 330, 360), pH = c(7.5, 5.0, 5.0, 7.5))
plot(pH ~ time, data = pH_dat, type = 'l')
```



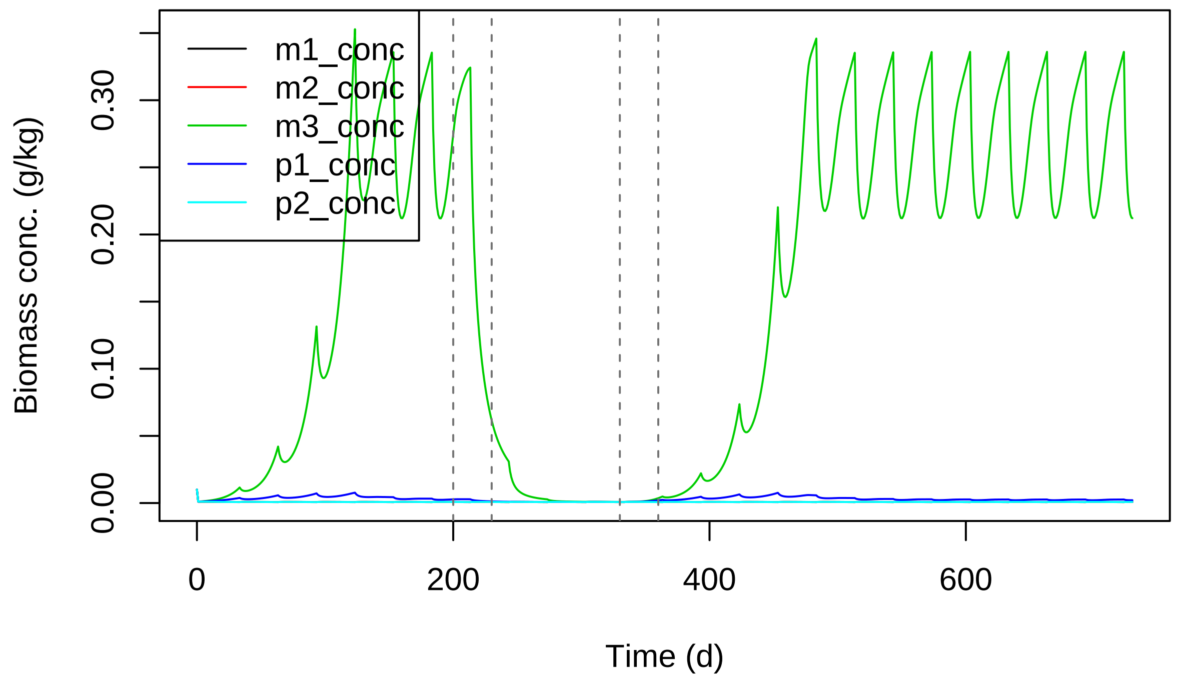
As with temperature, only the times that pH changes are needed–earlier and later times extend the nearest value.

```r
out8 <- abm(730, 1, add_pars = list(pH = pH_dat), approx_method_pH = 'linear')
```

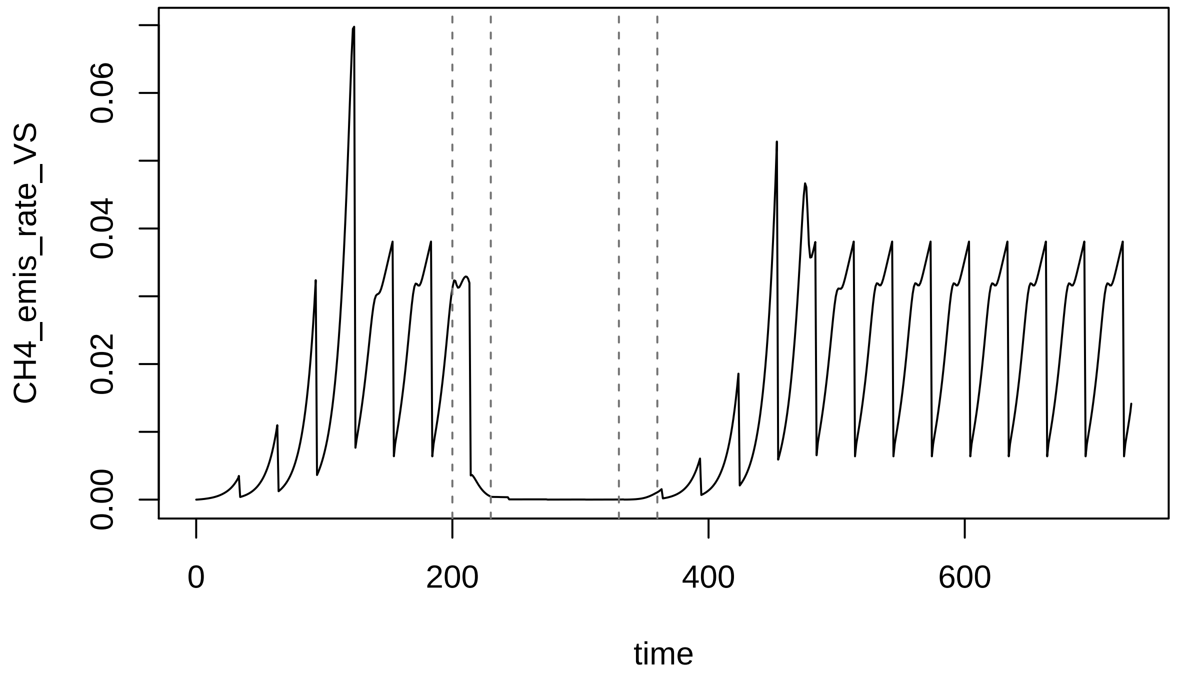
```r
plot(pH ~ time, type = 'l', col = 'red', data = out8)
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



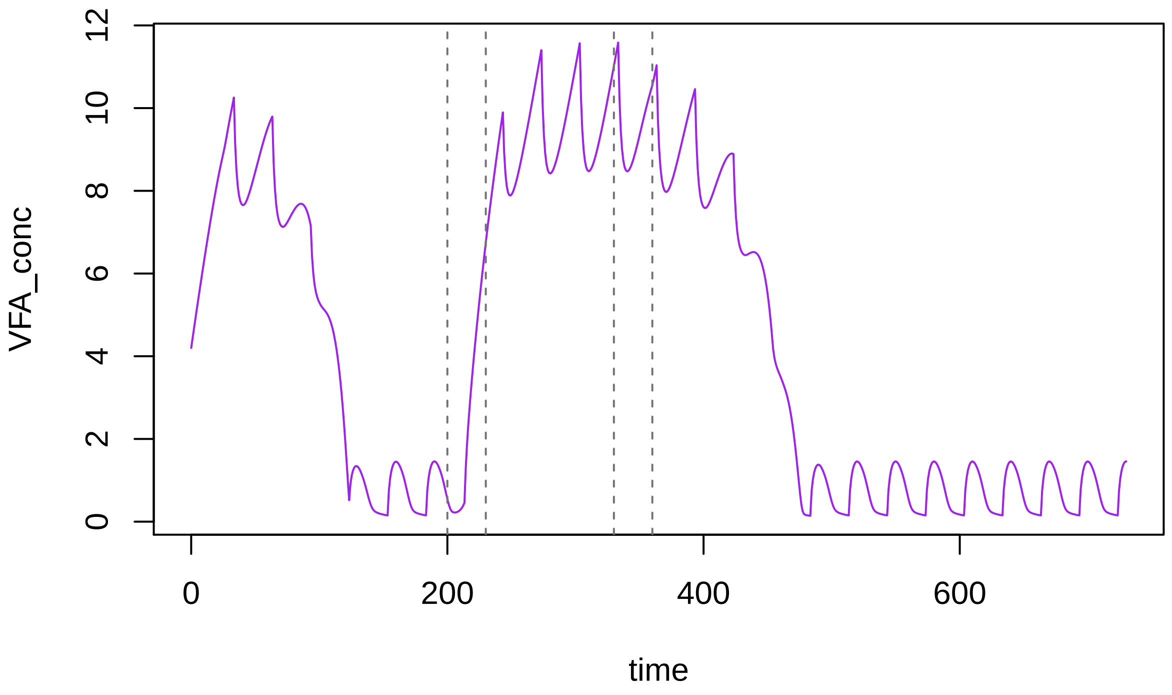
```r
matplot(out8$time, out8[, nn <- c('m1_conc', 'm2_conc', 'm3_conc', 'p1_conc', 'p2_conc')],
        type = 'l', lty = 1, xlab = 'Time (d)', ylab = 'Biomass conc. (g/kg)')
legend('topleft', nn, col = 1:5, lty = 1)
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



```
plot(CH4_emis_rate_VS ~ time, data = out8, type = 'l')
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



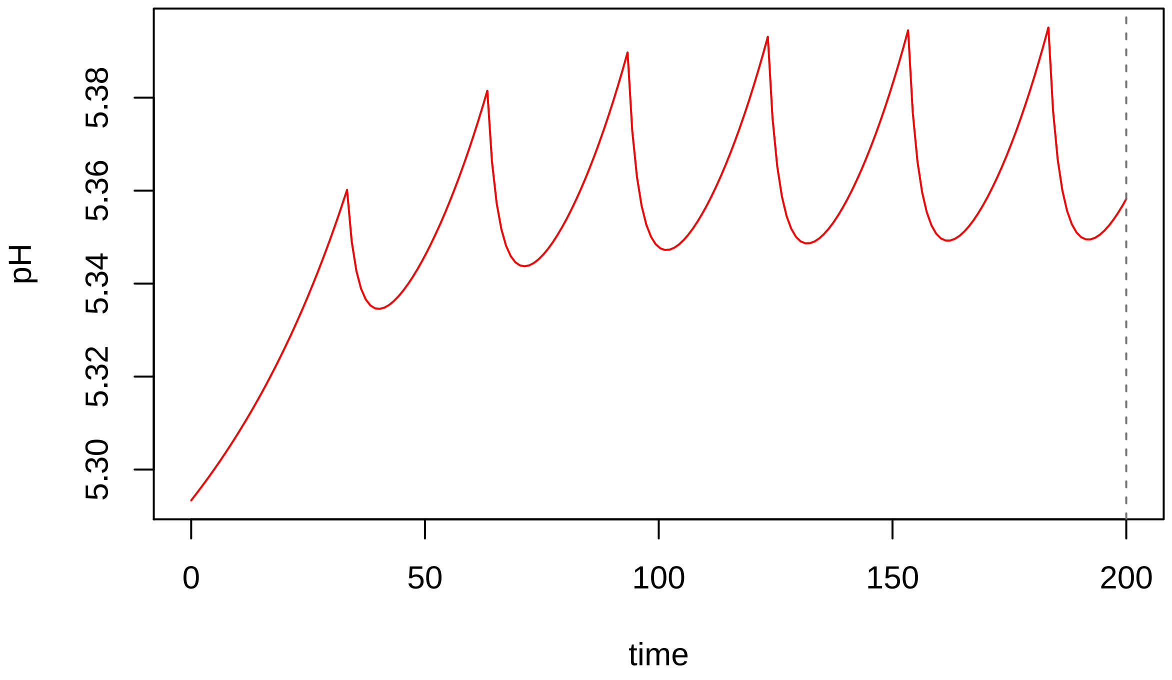
```
plot(VFA_conc ~ time, type = 'l', col = 'purple', data = out8)
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



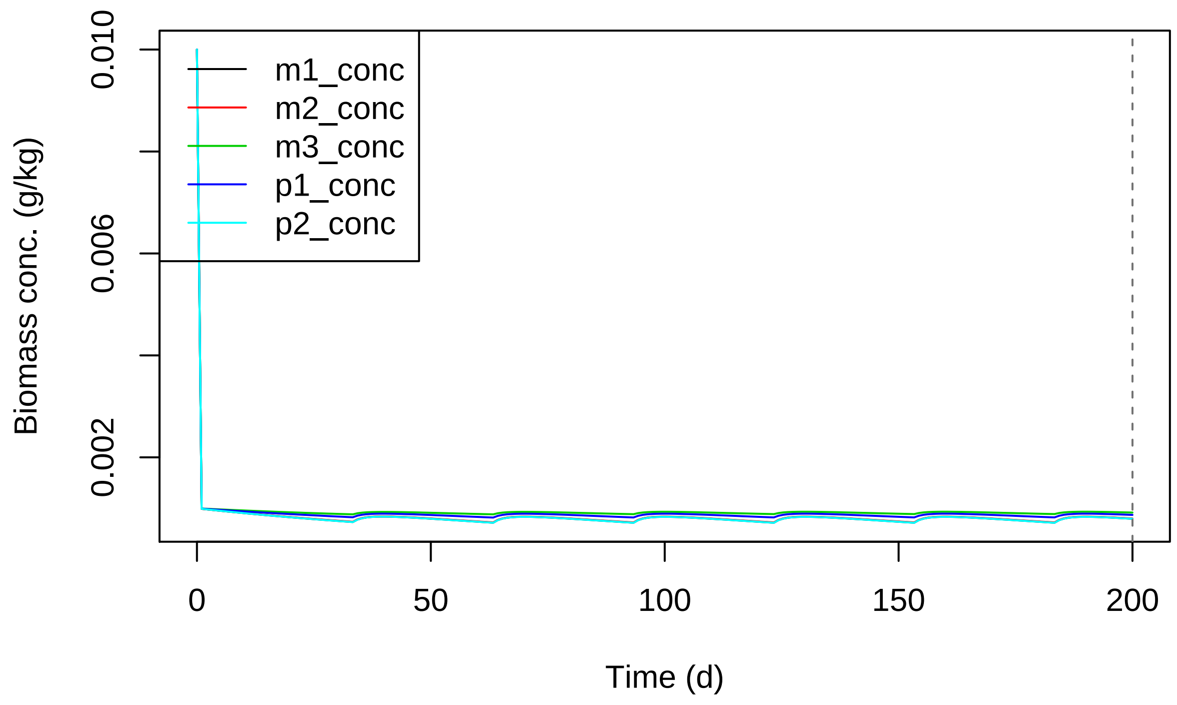
Alternatively, pH can be calculated from the H_2SO_4 concentration (given as $SO_4{}^{-2}$). This approach is based on a titration curve for "typical" slurry.

```
out9 <- abm(200, 1, add_pars = list(conc_fresh.SO4 = 1.3, pH = 'calc'))
```

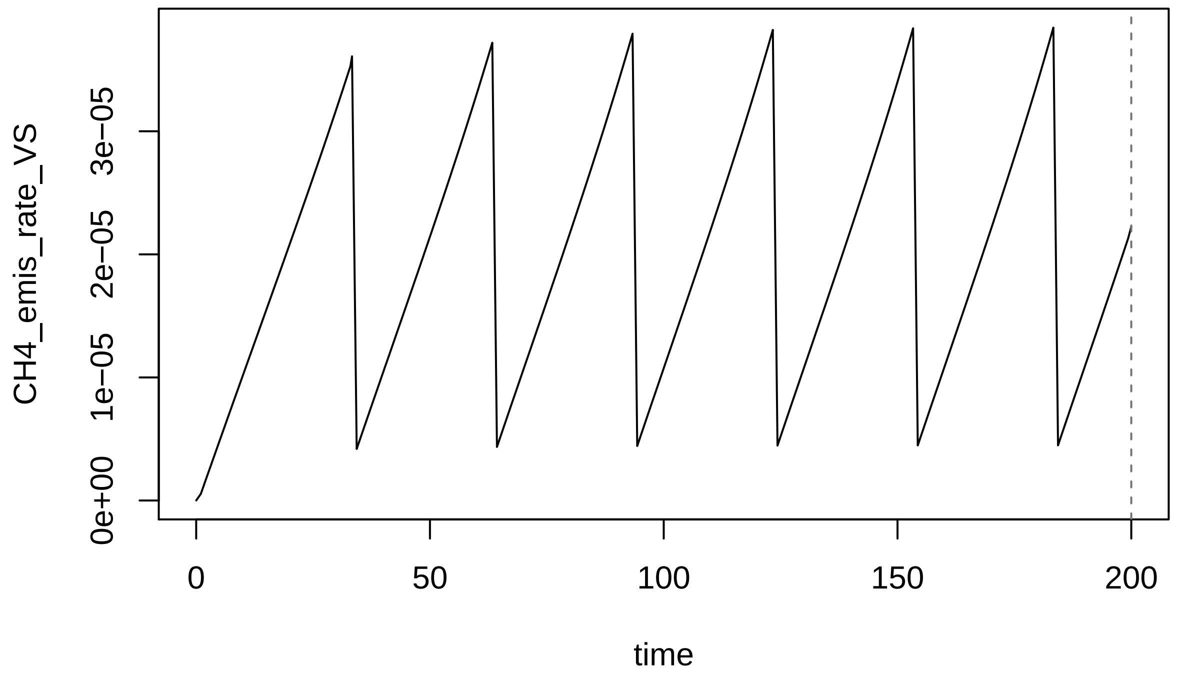
```
plot(pH ~ time, type = 'l', col = 'red', data = out9)
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



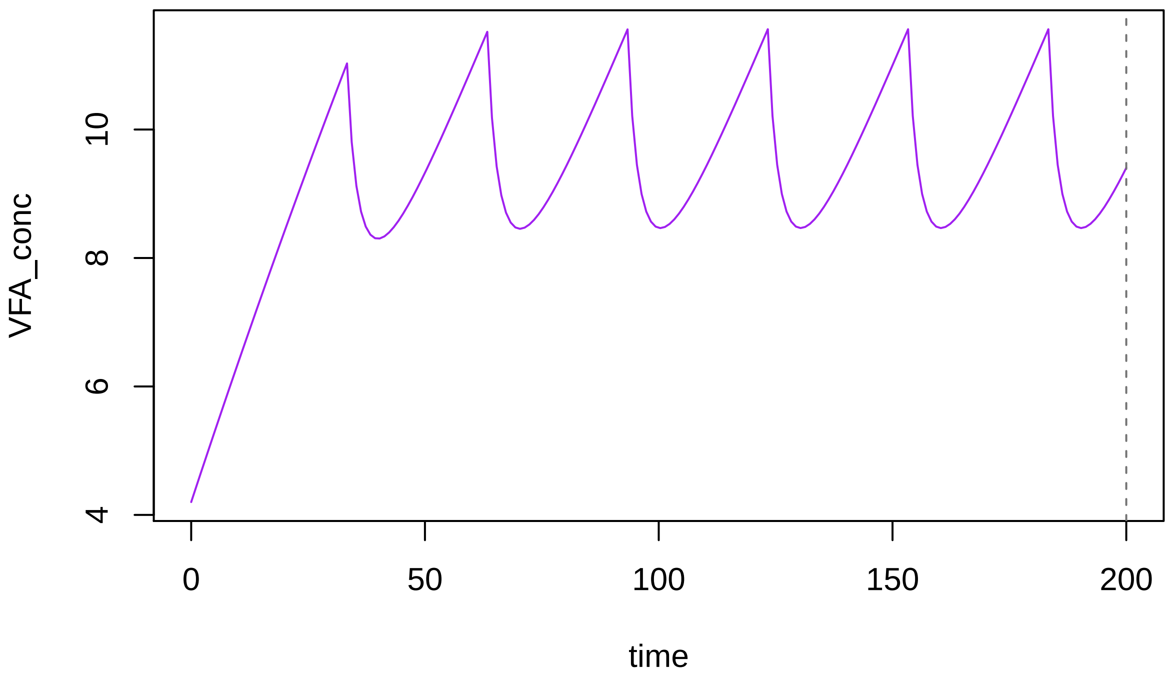
```
matplot(out9$time, out9[, nn <- c('m1_conc', 'm2_conc', 'm3_conc', 'p1_conc', 'p2_conc')],
        type = 'l', lty = 1, xlab = 'Time (d)', ylab = 'Biomass conc. (g/kg)')
legend('topleft', nn, col = 1:5, lty = 1)
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



```r
plot(CH4_emis_rate_VS ~ time, data = out9, type = 'l')
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



```r
plot(VFA_conc ~ time, type = 'l', col = 'purple', data = out9)
abline(v = pH_dat$time, lty = 2, col = 'gray45')
```



NTS: This is a difficult approach and the titration curve needs another look.

9. Model output as input

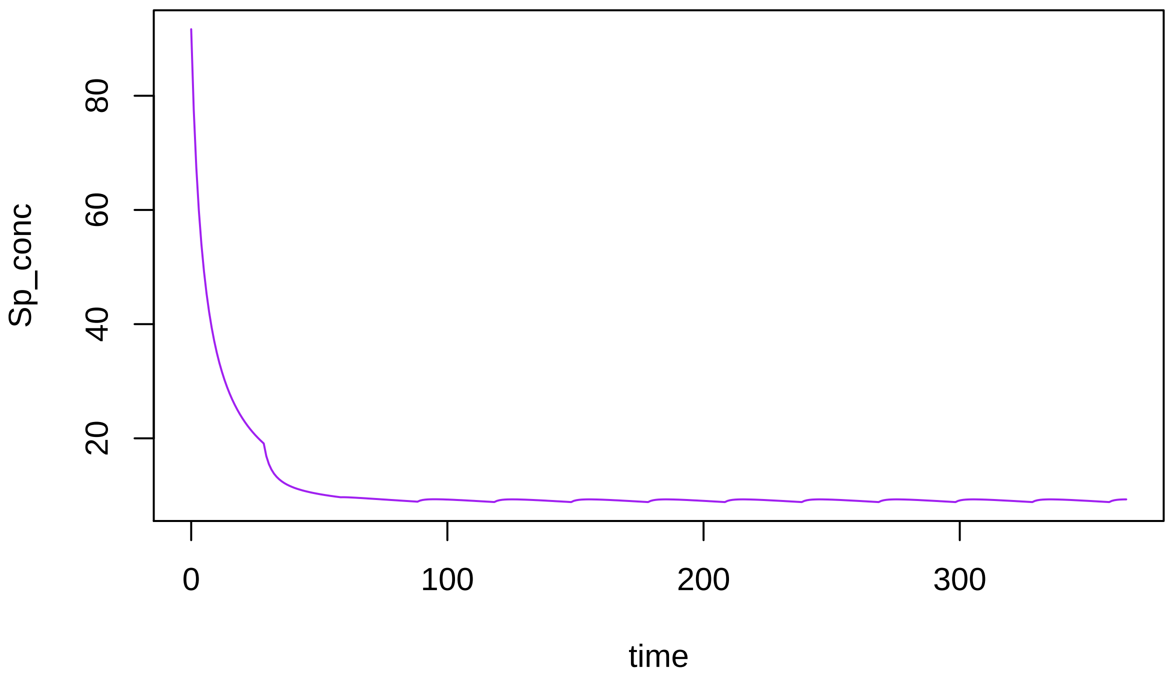
Although the `abm()` function is quite flexible and can accommodate changes in inputs over time, there is plenty it cannot do. To partially address limitations, it is possible to specify output from one call as the starting conditions for a new call. This is done using the `starting` argument. In the following example, two calls are used to show the effect of an instant change in the substrate concentration in fresh slurry.

```
out10 <- abm(365, 1, add_pars = list(conc_fresh.Sp = 100, conc_fresh.COD = 200, conc_fresh.VFA = 10))
```

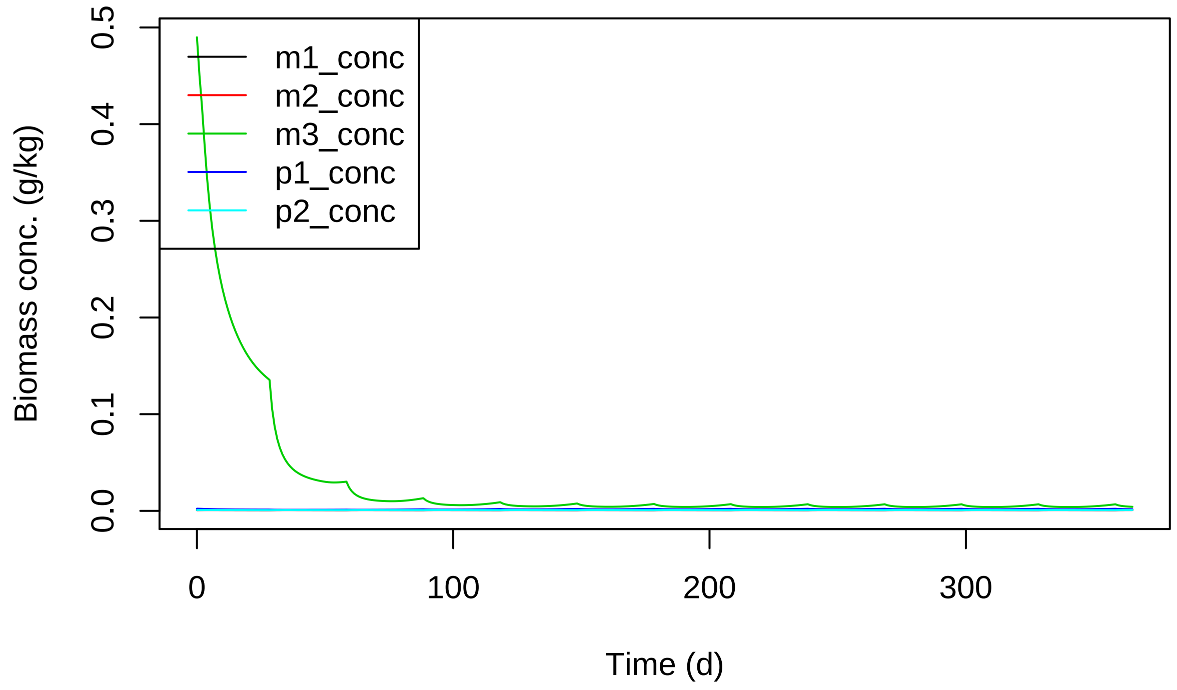
```
out11 <- abm(365, 1, add_pars = list(conc_fresh.Sp = 10, conc_fresh.COD = 20, conc_fresh.VFA = 0),
             starting = out10)
```

```
## Using starting conditions from `starting` argument
```

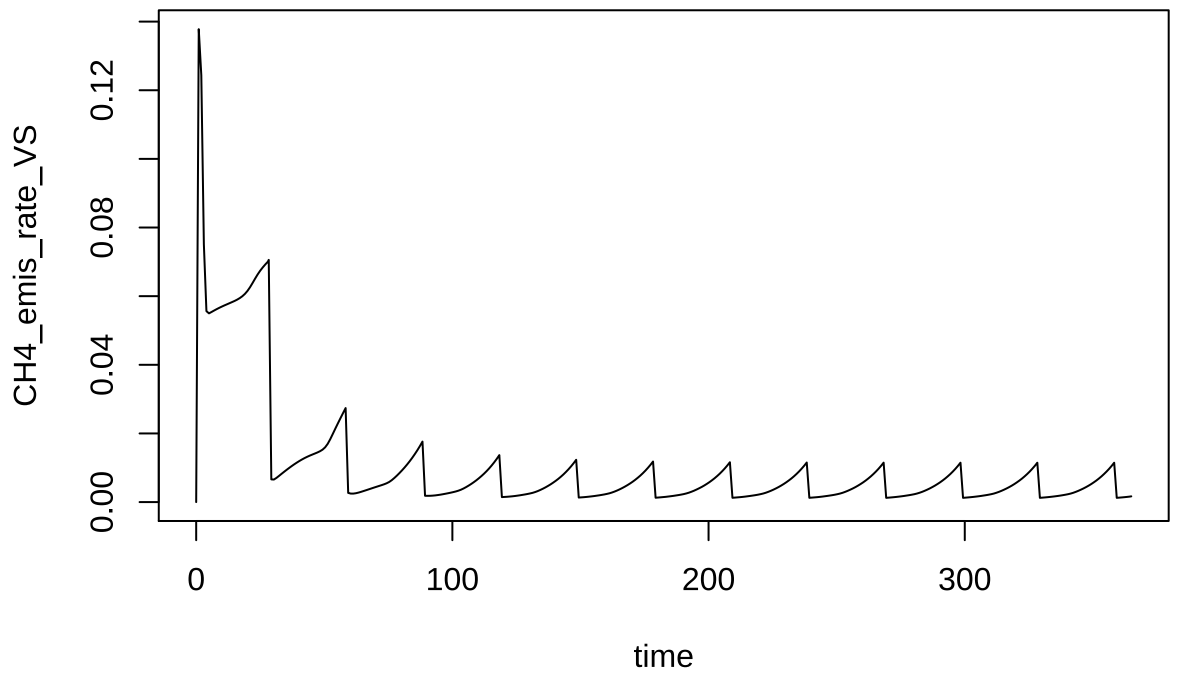
```
plot(Sp_conc ~ time, type = 'l', col = 'purple', data = out11)
```



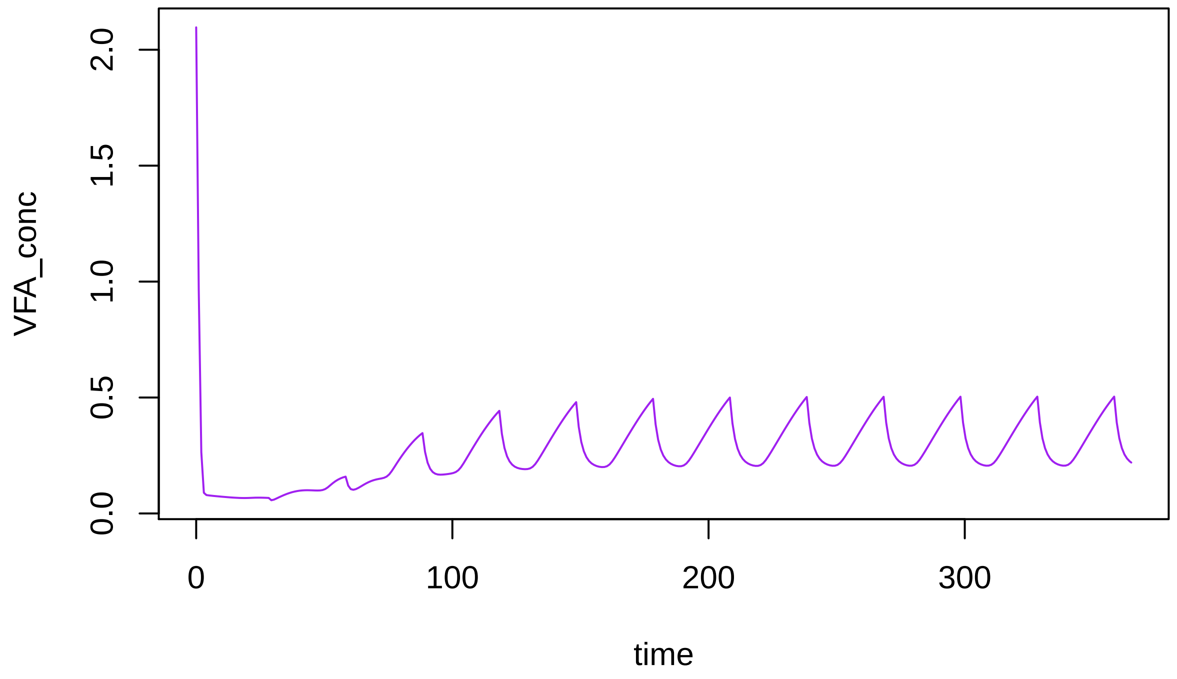
```
matplot(out11$time, out11[, nn <- c('m1_conc', 'm2_conc', 'm3_conc', 'p1_conc', 'p2_conc')],
        type = 'l', lty = 1, xlab = 'Time (d)', ylab = 'Biomass conc. (g/kg)')
legend('topleft', nn, col = 1:5, lty = 1)
```



```
plot(CH4_emis_rate_VS ~ time, data = out11, type = 'l')
```



```
plot(VFA_conc ~ time, type = 'l', col = 'purple', data = out11)
```



10. More information

Users can track development of the ABM package on GitHub: https://github.com/sashahafner/ABM. To report bugs or request features, use the “Issues” page. For information about the model, see the references listed below.

References