Demo of new simple1 version

Sasha D. Hafner

06 June, 2025 05:24

Overview

This demo shows:

1. basic usage,
2. variable substrates,
3. time-variable inputs,
4. speciation (proton-transfer reactions),
5. inhibition,
6. COD balance,

Prep

```r
devtools::load_all()
```

```
## i Loading ABM
```

Function arguments

The argument list currently looks like this:

```r
abm <- function(
  days = 365,                 # Number of days to run
  delta_t = 1,                # Time step for output
  times = NULL,               # Optional vector of times for output
  mng_pars,
  man_pars,
  init_pars = list(conc_init =  c(man_pars$comp_fresh, man_pars$VFA_fresh)),
  grp_pars,
  mic_pars,
  sub_pars,
  chem_pars,
  inhib_pars = NULL,
  mt_pars = NULL,
  ctrl_pars = list(respir = TRUE,
                   pH_inhib = FALSE,
                   approx_method = 'early',
                   par_key = '\\.',
                   rates_calc = 'instant'),
  var_pars = list(var = NULL),
  add_pars = NULL,
  pars = NULL,
  startup = 0,                 # Number of times complete simulation should be run before returning resul
  starting = NULL,             # Output from previous simulation to be starting condition for new one
  value = 'ts',                # Type of output
  warn = TRUE) {
```

The main changes in arguments are:

- new `sub_pars` for defining substrates
- new `ctrl_pars` for some "control" parameters
- new `var_pars` for *any* parameters that change over time

I have removed the default `*_pars` objects for now.

1. Basic behavior

The simplest usage is with constant slurry production rate and a fixed schedule. We need to set some parameters, first management parameters.

```r
mng_pars = list(slurry_prod_rate = 10000,
                slurry_mass = 1000,
                storage_depth = 2,
                resid_depth = 0.1,
                area = 100,
                empty_int = 100,
                temp_C = 20,
                wash_water = 0,
                wash_int = NA,
                rest_d = 0,
                resid_enrich = 1)
```

Next substrate parameters, a new argument. This defines substrates. We could have any number with any names. Note that hydrolysis uses CTM again (like anything here, that could be changed).

```r
sub_pars <- list(subs = c('VSd'),
                 T_opt_hyd = c(VSd = 60),
                 T_min_hyd = c(VSd = 0),
                 T_max_hyd = c(VSd = 90),
                 hydrol_opt = c(VSd = 0.1),
                 sub_fresh = c(VSd = 50),
                 sub_init = c(VSd = 50))
```

Microbial parameters are similar to other ABM versions, but inhibition is set separately now.

```r
grp_pars <- list(grps = c('m0', 'm1', 'm2','sr1'),
                 yield = c(default = 0.05, sr1 = 0.065),
                 xa_fresh = c(all = 0.05),
                 xa_init = c(all = 0.05),
                 dd_rate = c(all = 0.02),
                 ks = c(default = 1, sr1 = 0.5),
                 qhat_opt = c(m0 = 1, m1 = 1, m2 = 2, sr1 = 9),
                 T_opt = c(m0 = 18, m1 = 18, m2 = 28, sr1 = 44),
                 T_min = c(m0 = 0, m1 = 6.41, m2 = 6.41, sr1 = 0),
                 T_max = c(m0 = 25, m1 = 25, m2 = 38, sr1 = 51))
```

The `dd_rate_xa` parameter is for "death and decay".

```r
mic_pars <- list(dd_rate_xa = 0.02)
```

These last two arguments are similar to other versions. VFA is hard-wired and so has its own elements.

```
man_pars <- list(VFA_fresh = c(VFA = 2), pH = 7, dens = 1000)

chem_pars <- list(COD_conv = c(CH4 = 1/0.2507, xa = 1/0.7069561,
                               VFA = 1/0.9383125, S = 1/0.5015, VS = 1/0.69,
                               CO2_aer = 1/0.436, CO2_sr = 1/1.2,
                               C_xa = 1/0.3753125))
```

```
out1 <- abm(365,
            mng_pars = mng_pars,
            man_pars = man_pars,
            grp_pars = grp_pars,
            mic_pars = mic_pars,
            sub_pars = sub_pars,
            chem_pars = chem_pars)
```

```
## Warning in checkCOD(dat = dat, grps = pars$grps, subs = pars$subs, COD_conv =
## pars$COD_conv, : COD balance is off by 1.7%
```

Output is similar to other versions. (The `value` argument does not currently work.)

```
head(out1)
```

```
##   time        m0        m1        m2       sr1       VSd       VFA slurry_mass
## 1    0   50.0000   50.0000   50.0000   50.0000   50000.0   2000.00        1000
## 2    1  554.0098  553.8533  558.3748  544.0431  542318.4  29018.66       11000
## 3    2 1066.2767 1065.6732 1083.1940 1028.3035 1022076.9  67371.44       21000
## 4    3 1588.3161 1586.9430 1627.0197 1502.9749 1489597.5 116611.44       31000
## 5    4 2121.2034 2118.7114 2191.8594 1968.2472 1945194.0 176326.70       41000
## 6    5 2665.7726 2661.7877 2779.4361 2424.3064 2389172.2 246127.08       51000
##   CH4_emis_cum CO2_emis_cum slurry_load COD_load CH4_emis_rate temp_C pH m0_eff
## 1       0.0000       0.0000           0        0      25.52844     20  7      0
## 2     163.6272     130.5362       10000   522000     308.65512     20  7      0
## 3     628.8119     501.6449       20000  1044000     626.59326     20  7      0
## 4    1425.4337    1137.1629       30000  1566000     970.52564     20  7      0
## 5    2577.0208    2055.8603       40000  2088000    1335.99741     20  7      0
## 6    4103.8067    3273.8785       50000  2610000    1720.64113     20  7      0
##   m1_eff m2_eff sr1_eff VSd_eff VFA_eff slurry_mass_eff slurry_depth    m0_conc
## 1      0      0       0       0       0               0         0.01 0.05000000
## 2      0      0       0       0       0               0         0.11 0.05036453
## 3      0      0       0       0       0               0         0.21 0.05077508
## 4      0      0       0       0       0               0         0.31 0.05123600
## 5      0      0       0       0       0               0         0.41 0.05173667
## 6      0      0       0       0       0               0         0.51 0.05227005
##      m1_conc    m2_conc   sr1_conc VSd_conc VFA_conc m0_eff_conc m1_eff_conc
## 1 0.05000000 0.05000000 0.05000000 50.00000 2.000000         NaN         NaN
## 2 0.05035030 0.05076135 0.04945846 49.30167 2.638060         NaN         NaN
## 3 0.05074634 0.05158067 0.04896683 48.67033 3.208164         NaN         NaN
## 4 0.05119171 0.05248451 0.04848306 48.05153 3.761660         NaN         NaN
## 5 0.05167589 0.05345999 0.04800603 47.44376 4.300651         NaN         NaN
## 6 0.05219192 0.05449875 0.04753542 46.84651 4.826021         NaN         NaN
##   m2_eff_conc sr1_eff_conc VSd_eff_conc VFA_eff_conc
## 1         NaN          NaN          NaN          NaN
## 2         NaN          NaN          NaN          NaN
## 3         NaN          NaN          NaN          NaN
## 4         NaN          NaN          NaN          NaN
## 5         NaN          NaN          NaN          NaN
## 6         NaN          NaN          NaN          NaN
```

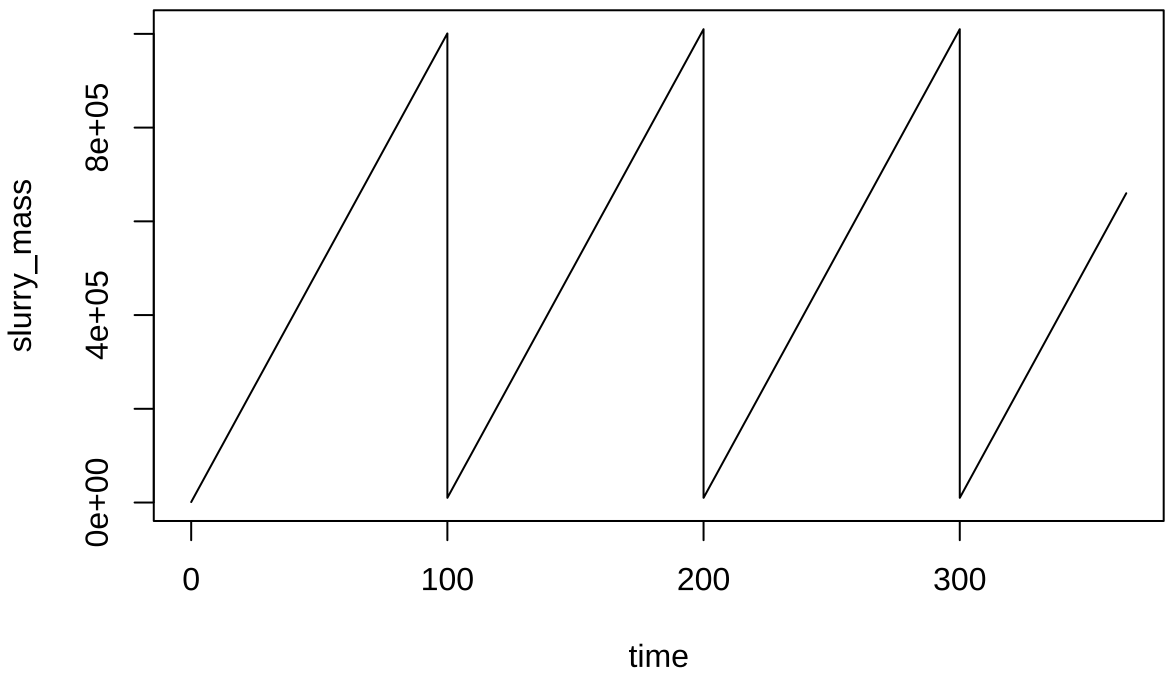
```r
tail(out1)
```

```
##     time       m0       m1       m2      sr1      VSd     VFA slurry_mass
## 364  360 74909.98 72497.35 337403.6 17643.06 15468816 4796289      610000
## 365  361 77056.84 74543.26 351549.2 17788.74 15576604 4684393      620000
## 366  362 79234.75 76617.42 366160.3 17931.53 15682002 4557322      630000
## 367  363 81441.36 78717.58 381233.0 18071.50 15785084 4414981      640000
## 368  364 83673.78 80840.93 396758.5 18208.69 15885918 4257395      650000
## 369  365 85928.37 82984.01 412722.1 18343.17 15984571 4084743      660000
##     CH4_emis_cum CO2_emis_cum slurry_load  COD_load CH4_emis_rate temp_C pH
## 364     26846605     21417315     3600000 187920000      125134.1     20  7
## 365     26973866     21518840     3610000 188442000      129394.2     20  7
## 366     27105400     21623774     3620000 188964000      133674.8     20  7
## 367     27241214     21732122     3630000 189486000      137950.5     20  7
## 368     27381289     21843868     3640000 190008000      142189.3     20  7
## 369     27525568     21958969     3650000 190530000      146351.3     20  7
##       m0_eff   m1_eff  m2_eff  sr1_eff  VSd_eff VFA_eff slurry_mass_eff
## 364 441740.7 422210.1 2239891 63286.93 53531054 3419880         2991000
## 365 441740.7 422210.1 2239891 63286.93 53531054 3419880         2991000
## 366 441740.7 422210.1 2239891 63286.93 53531054 3419880         2991000
## 367 441740.7 422210.1 2239891 63286.93 53531054 3419880         2991000
## 368 441740.7 422210.1 2239891 63286.93 53531054 3419880         2991000
## 369 441740.7 422210.1 2239891 63286.93 53531054 3419880         2991000
##     slurry_depth   m0_conc   m1_conc   m2_conc   sr1_conc VSd_conc VFA_conc
## 364          6.1 0.1228032 0.1188481 0.5531206 0.02892305 25.35872 7.862769
## 365          6.2 0.1242852 0.1202311 0.5670148 0.02869152 25.12355 7.555472
## 366          6.3 0.1257694 0.1216150 0.5812069 0.02846275 24.89207 7.233845
## 367          6.4 0.1272521 0.1229962 0.5956766 0.02823671 24.66419 6.898408
## 368          6.5 0.1287289 0.1243707 0.6103976 0.02801337 24.43987 6.549838
## 369          6.6 0.1301945 0.1257334 0.6253365 0.02779268 24.21905 6.189004
##     m0_eff_conc m1_eff_conc m2_eff_conc sr1_eff_conc VSd_eff_conc VFA_eff_conc
## 364     0.14769   0.1411602   0.7488771   0.02115912     17.89738      1.14339
## 365     0.14769   0.1411602   0.7488771   0.02115912     17.89738      1.14339
## 366     0.14769   0.1411602   0.7488771   0.02115912     17.89738      1.14339
## 367     0.14769   0.1411602   0.7488771   0.02115912     17.89738      1.14339
## 368     0.14769   0.1411602   0.7488771   0.02115912     17.89738      1.14339
## 369     0.14769   0.1411602   0.7488771   0.02115912     17.89738      1.14339
```

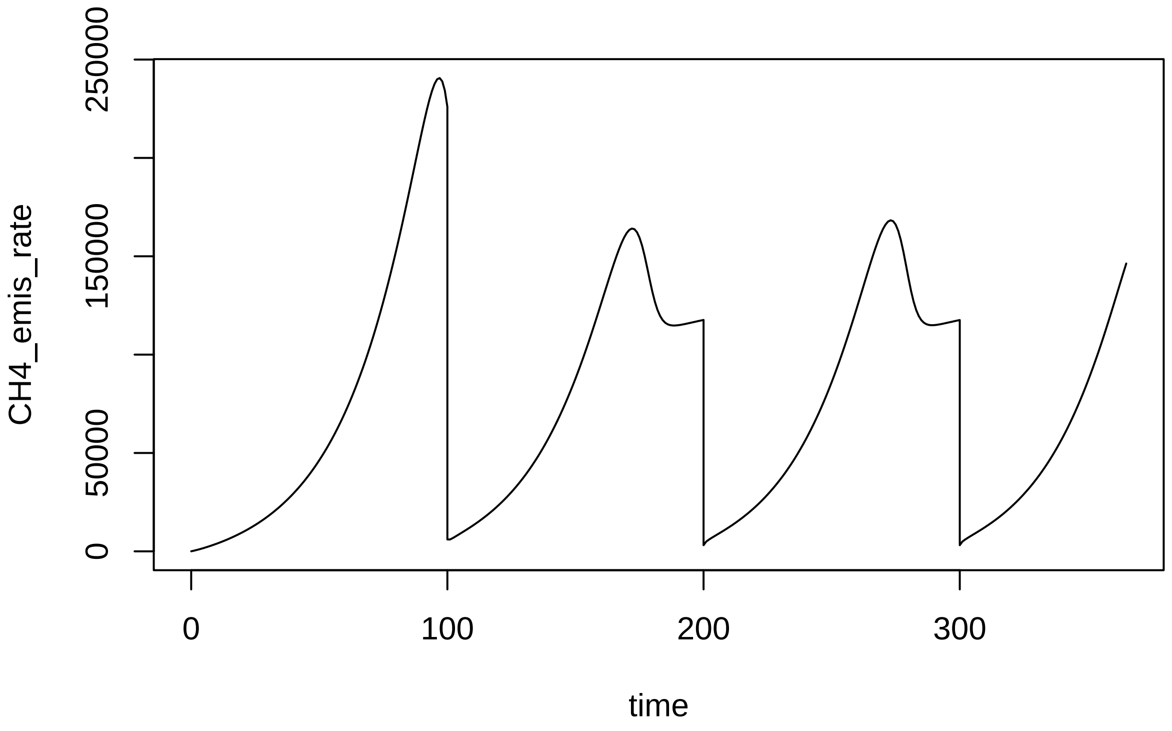
The effluent columns are cumulative. Is this what we had before? I did it for COD balance checking. We will have to discuss what is needed.

Here are some results.

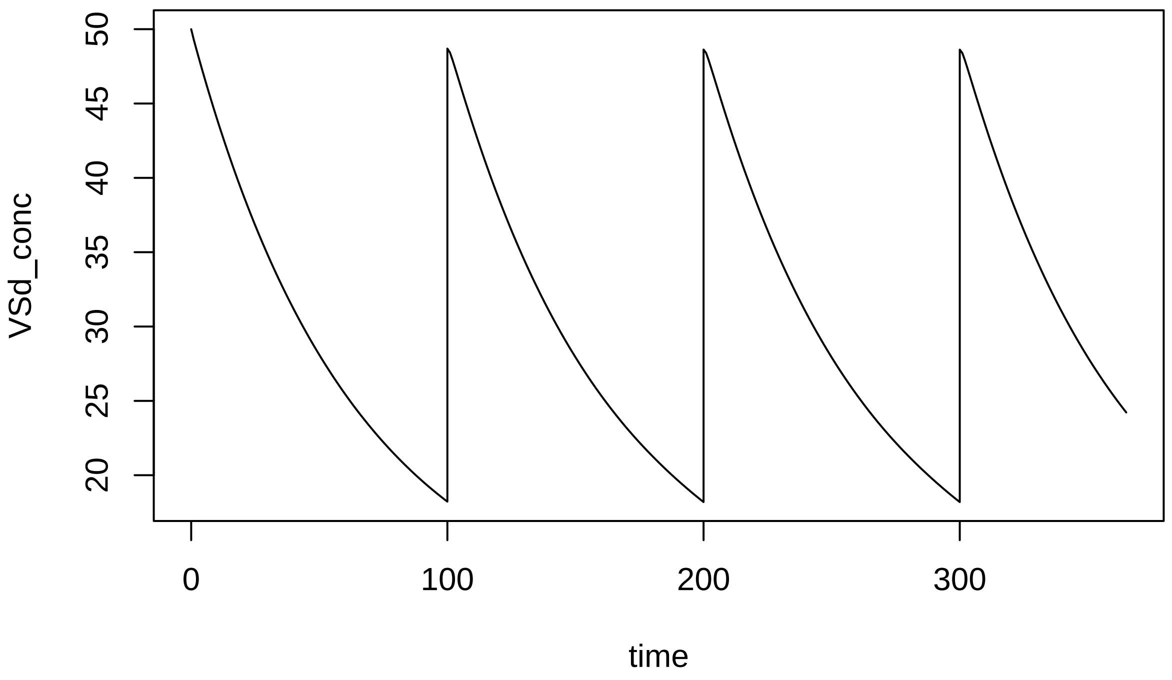
```r
plot(slurry_mass ~ time, data = out1, type = 'l')
```



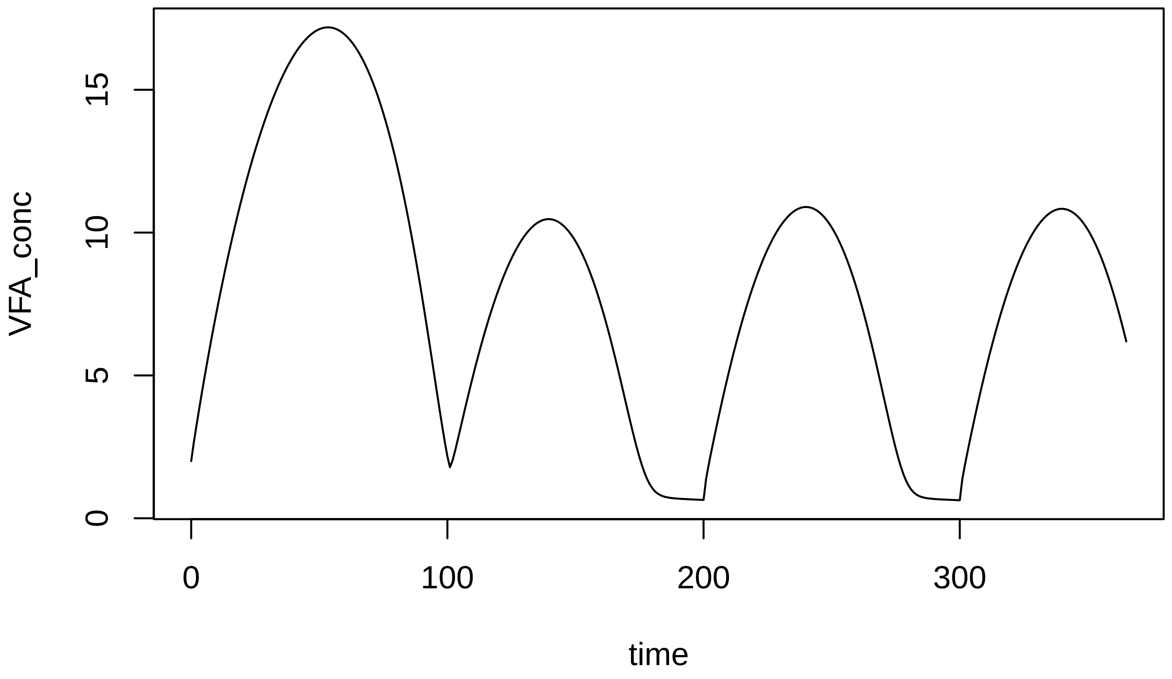
```r
plot(CH4_emis_rate ~ time, data = out1, type = 'l')
```



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

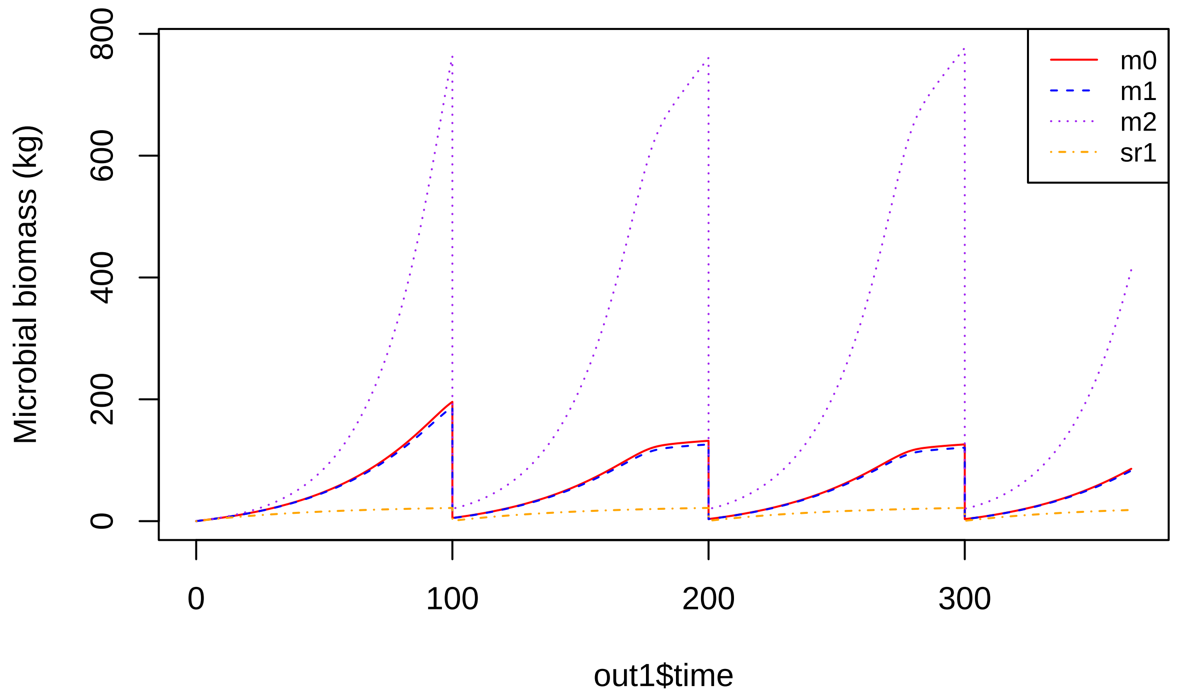


```r
plot(VFA_conc ~ time, data = out1, type = 'l')
```



And methanogens.

```r
line_colors <- c('red', 'blue', 'purple','orange')
matplot(out1$time, out1[, nn <- c('m0','m1','m2','sr1')]/1000,
        type = 'l', lty = c(1:length(nn)), col = line_colors, ylab = 'Microbial biomass (kg)')
legend("topright", legend = nn, lty = c(1:length(nn)), col = line_colors, lwd = 1, cex = 0.8)
```



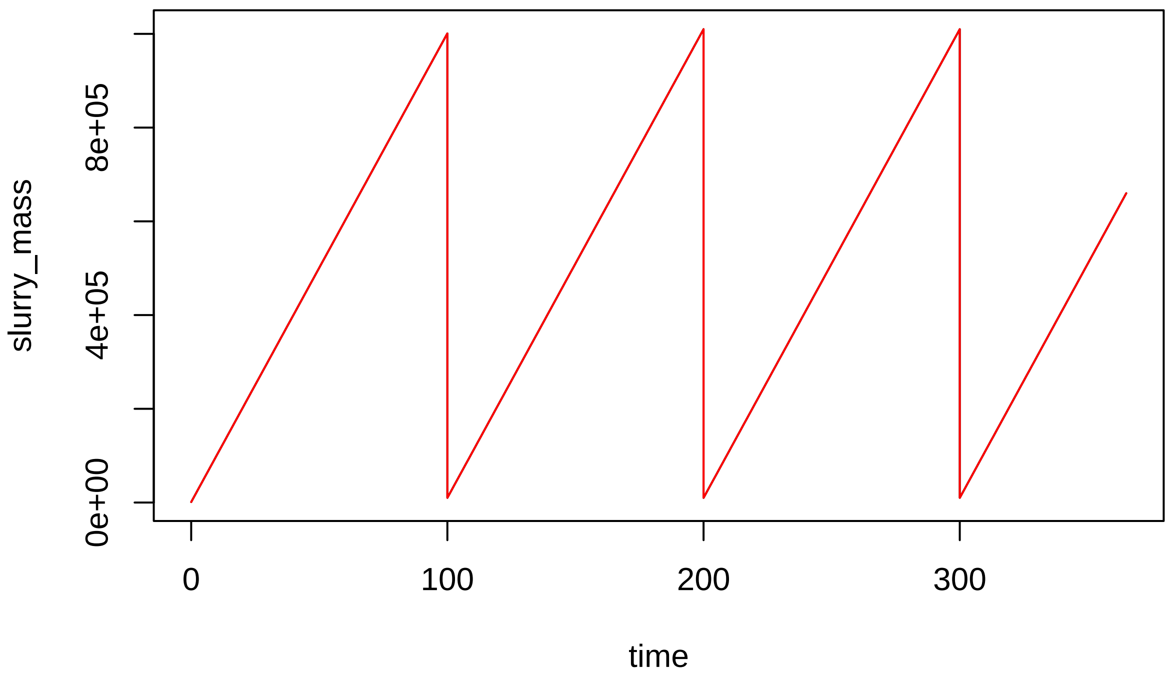
2. Substrate flexibility

Particulate substrates are defined in `sub_pars` now and there are no specific substrates hard-wired in the code. VFA is the only intermediate, and it is hard-wired. Here we will use three substrates. Parameter values have no connection to reality.

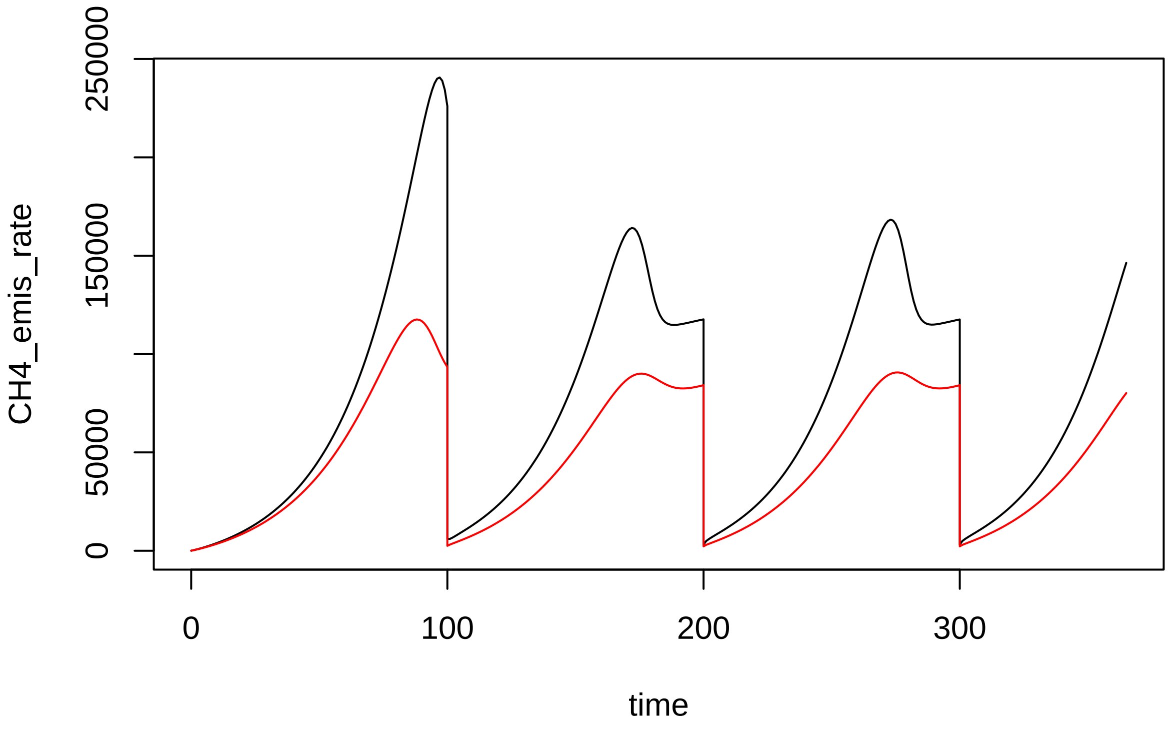
```
sub_pars2 <- list(subs = c('cellulose', 'protein', 'lipids'),
                  T_opt_hyd = c(all = 60),
                  T_min_hyd = c(all = 0),
                  T_max_hyd = c(all = 90),
                  hydrol_opt = c(lipids = 0.1, protein = 0.01, cellulose = 0.05),
                  sub_fresh = c(lipids = 3, protein = 20, cellulose = 35),
                  sub_init = c(lipids = 3, protein = 20, cellulose = 35))
```

```
out2 <- abm(365,
            mng_pars = mng_pars,
            man_pars = man_pars,
            grp_pars = grp_pars,
            mic_pars = mic_pars,
            sub_pars = sub_pars2,
            chem_pars = chem_pars)
```

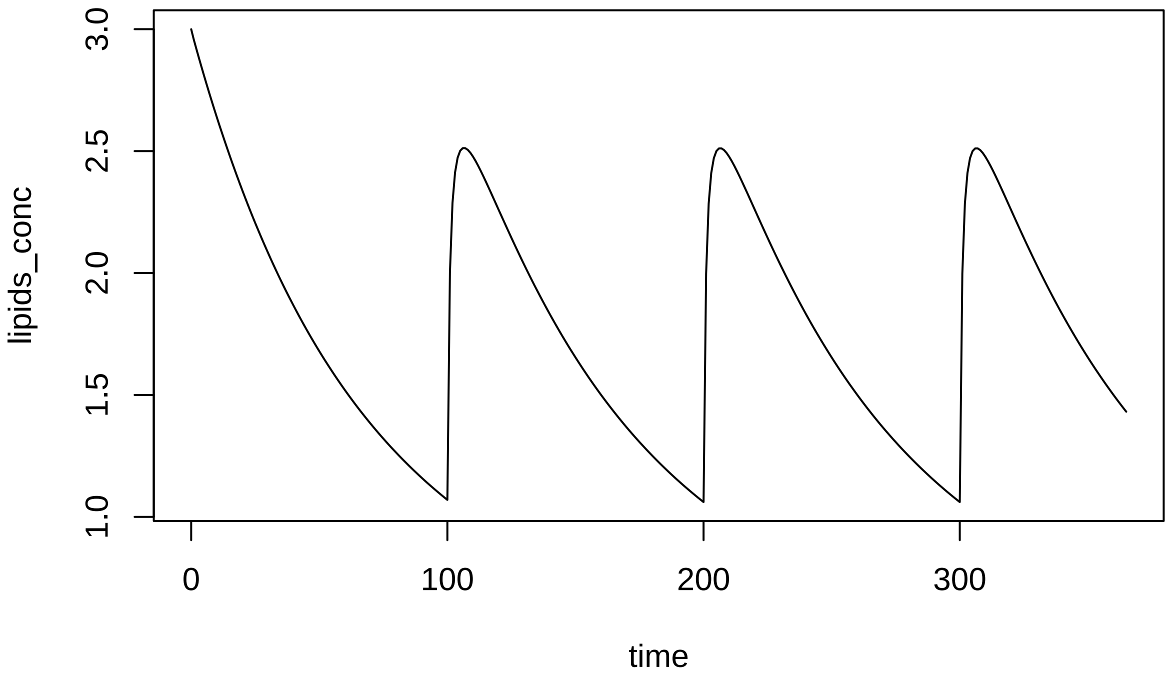
```
plot(slurry_mass ~ time, data = out2, type = 'l')
lines(slurry_mass ~ time, data = out1, col = 'red')
```



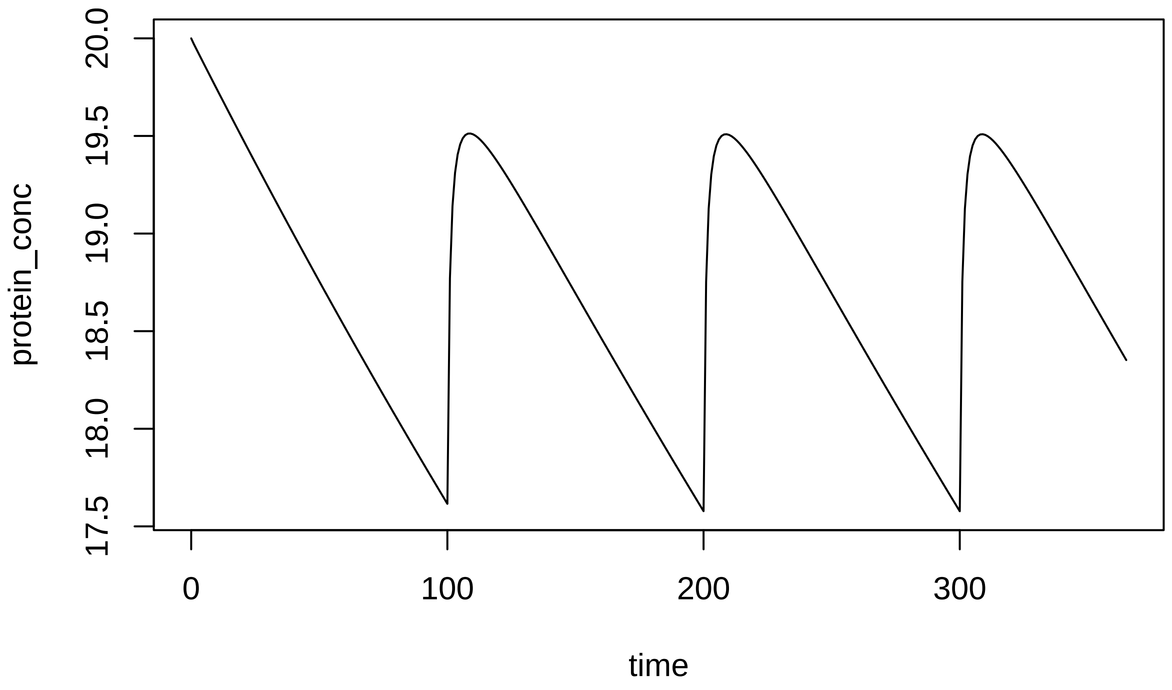
```
plot(CH4_emis_rate ~ time, data = out1, type = 'l')
lines(CH4_emis_rate ~ time, data = out2, col = 'red')
```



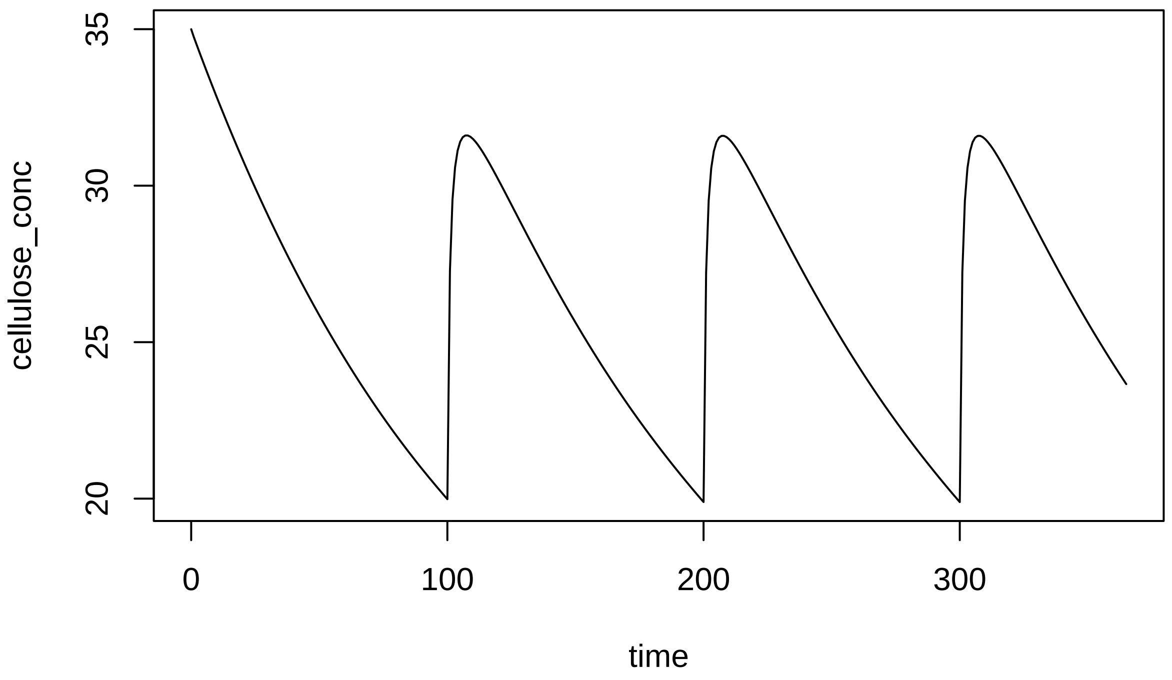
```
plot(lipids_conc ~ time, data = out2, type = 'l')
```



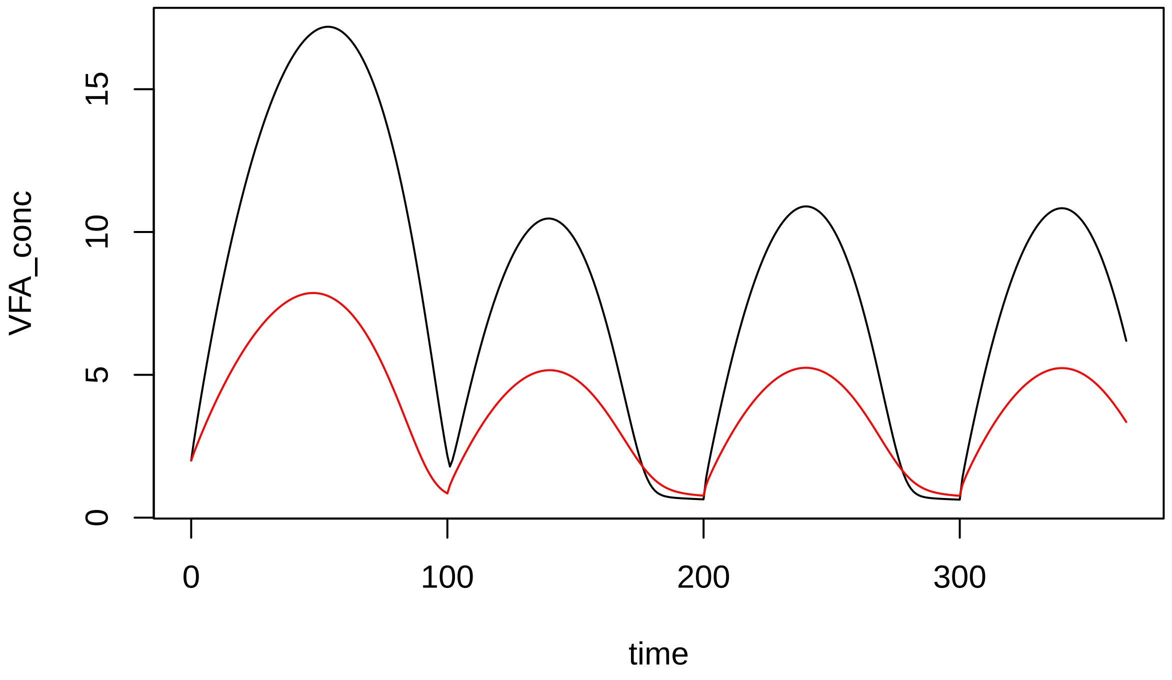
```r
plot(protein_conc ~ time, data = out2, type = 'l')
```



```r
plot(cellulose_conc ~ time, data = out2, type = 'l')
```



```r
plot(VFA_conc ~ time, data = out1, type = 'l')
lines(VFA_conc ~ time, data = out2, col = 'red')
```

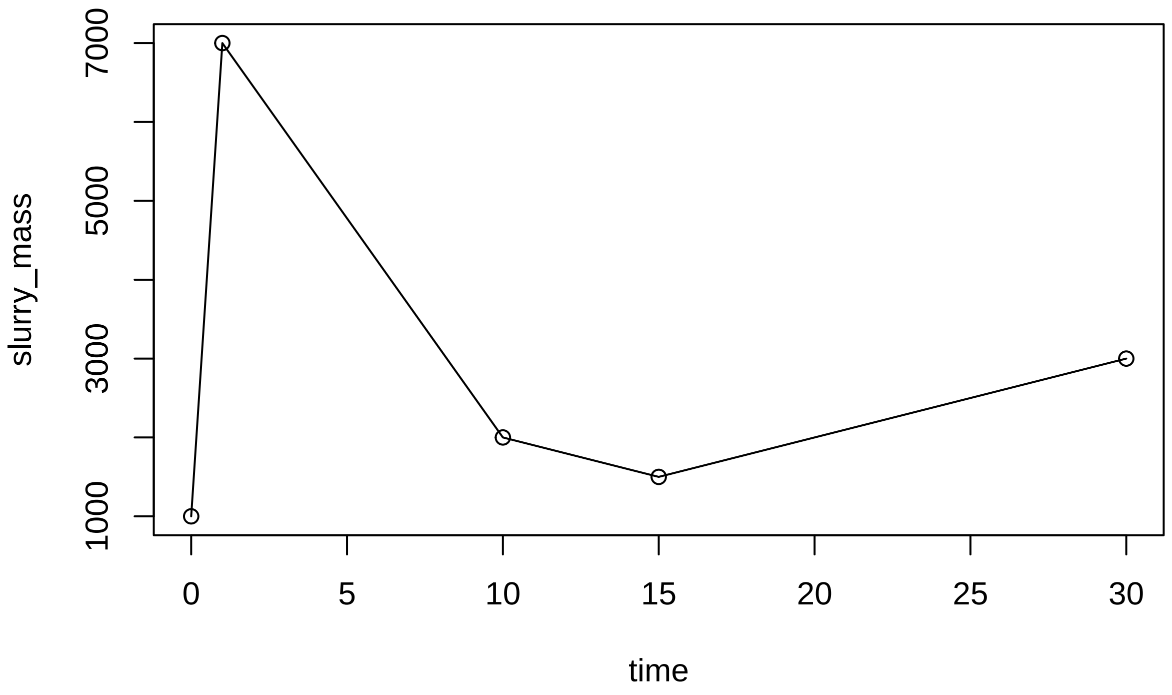


This flexibility comes from an approach similar to what we used for microbial groups.

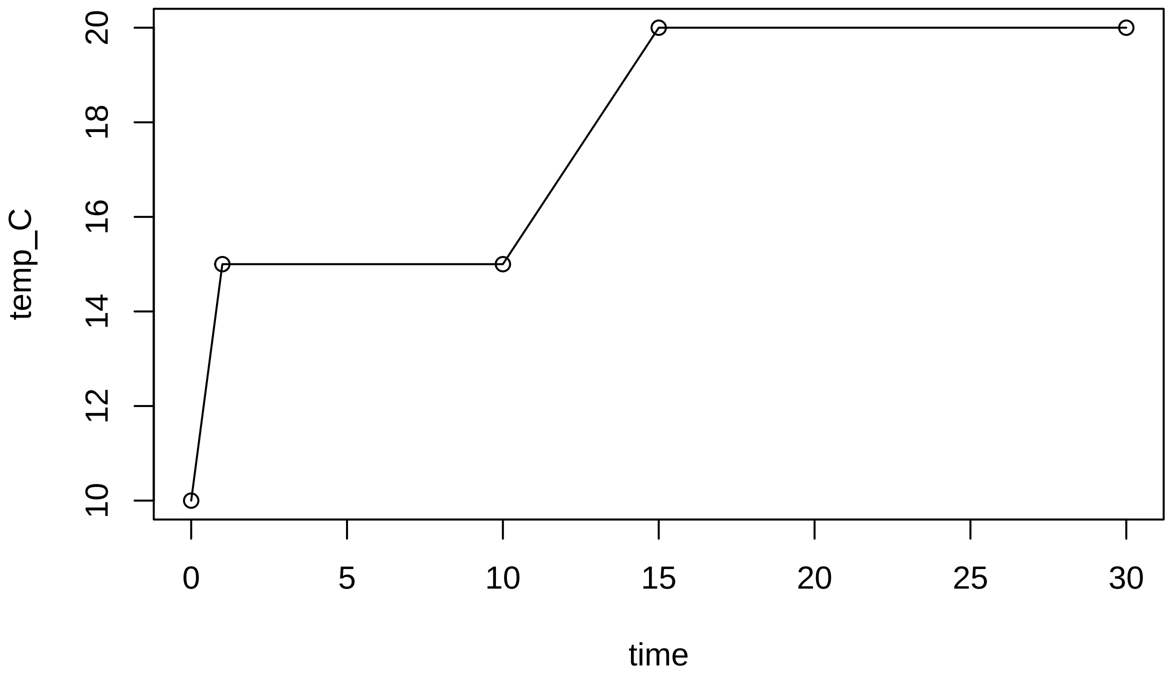
3. Time-variable inputs part 1

The `abm()` function can handle variability over time in any inputs now. Here slurry mass and temperature will vary.

```r
var_dat <- data.frame(time = c(0, 1, 10, 15, 30),
                      slurry_mass = c(1000, 7000, 2000, 1500, 3000),
                      temp_C = c(10, 15, 15, 20, 20))

plot(slurry_mass ~ time, data = var_dat, type = 'o')
```



```r
plot(temp_C ~ time, data = var_dat, type = 'o')
```

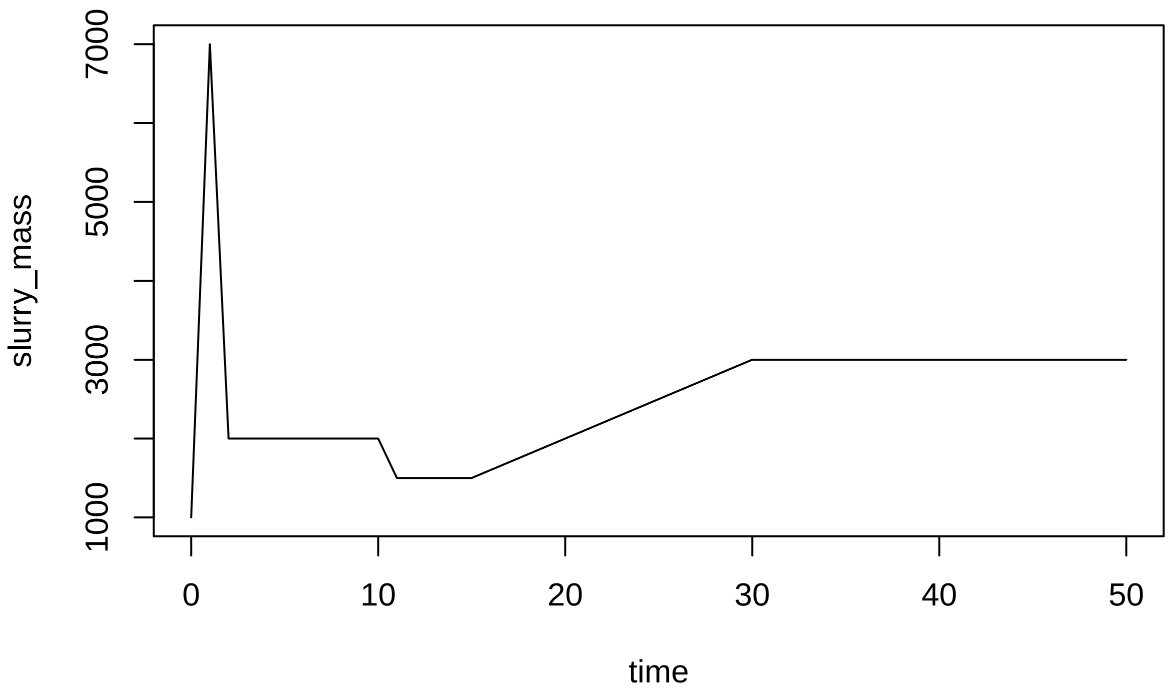


This data frame goes in the `var_pars` argument, which must be a list, even though it might have a single element named `var`. The `var` element is the only required one. The `var` data frame must have a `slurry_mass` column if it is used–it is not possible to use an `abm_regular()`-like approach with variable temperature etc.

```
var_pars <- list(var = var_dat)
```

```
out3a <- abm(50,
             mng_pars = mng_pars,
             man_pars = man_pars,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars,
             var_pars = var_pars)
```

```
plot(slurry_mass ~ time, data = out3a, type = 'l')
```

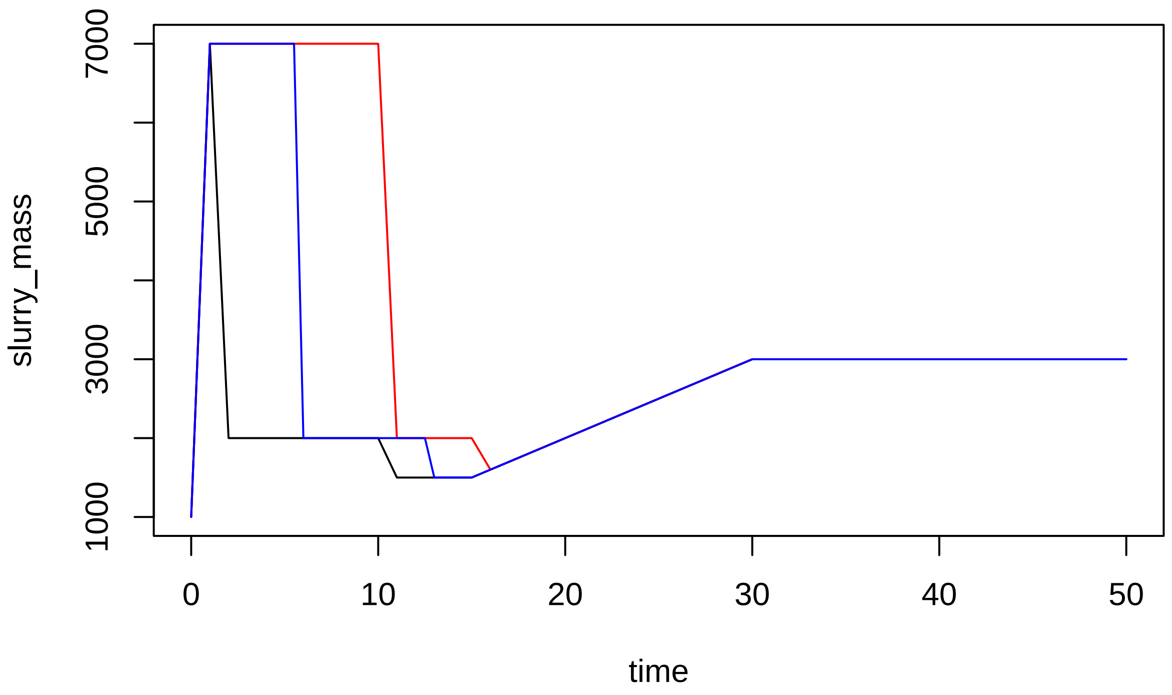


The “late” and “mid” options are still available, but now through `ctrl_pars`. Here we can change the value through `add_pars`

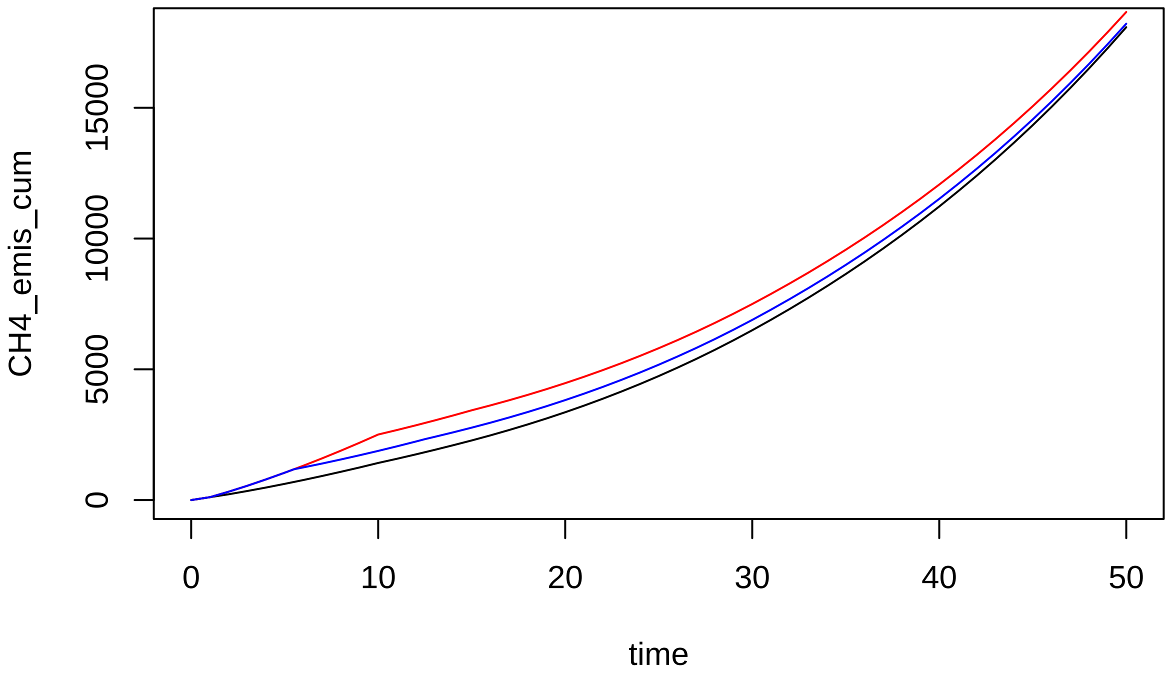
```
out3b <- abm(50,
             mng_pars = mng_pars,
             man_pars = man_pars,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars,
             var_pars = var_pars,
             add_pars = list(approx_method = 'late'))
```

```
out3c <- abm(50,
             mng_pars = mng_pars,
             man_pars = man_pars,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars,
             var_pars = var_pars,
             add_pars = list(approx_method = 'mid'))
```

```
plot(slurry_mass ~ time, data = out3a, type = 'l')
lines(slurry_mass ~ time, data = out3b, col = 'red')
lines(slurry_mass ~ time, data = out3c, col = 'blue')
```



```
plot(CH4_emis_cum ~ time, data = out3a, type = 'l')
lines(CH4_emis_cum ~ time, data = out3b, col = 'red')
lines(CH4_emis_cum ~ time, data = out3c, col = 'blue')
```



4. Time-variable inputs part 2

Here we'll vary fresh substrate concentrations over time.

First the data frame with slurry mass.

```
var_dat <- data.frame(time = c(0, 1, 10, 15, 30, 50),
                      slurry_mass = c(1000, 7000, 2000, 5000, 3000, 10000))
var_dat
```

```
##   time slurry_mass
## 1    0        1000
## 2    1        7000
## 3   10        2000
## 4   15        5000
## 5   30        3000
## 6   50       10000
```

Then add `sub_fresh` values. Each row needs a list containing a named vector. This is somewhat unusual data frame usage, and there is a user-friendly alternative based on additional data frames in the `var` argument (see next section). But I've kept this demo.

```
var_dat
```

```
##   time slurry_mass
## 1    0        1000
## 2    1        7000
## 3   10        2000
## 4   15        5000
## 5   30        3000
## 6   50       10000
```

```
var_dat$sub_fresh <- rep(list(c(VSd = 50)), nrow(var_dat))
var_dat$sub_fresh[3] <- list(c(VSd = 100))
var_dat$sub_fresh[4] <- list(c(VSd = 10))
var_dat$sub_fresh[5] <- list(c(VSd = 0))
var_dat$sub_fresh[6] <- list(c(VSd = 200))
var_dat
```

```
##   time slurry_mass sub_fresh
## 1    0        1000        50
## 2    1        7000        50
## 3   10        2000       100
## 4   15        5000        10
## 5   30        3000         0
## 6   50       10000       200
```

```
var_dat[1, 3]
```

```
## [[1]]
## VSd
##  50
```

```
var_dat[5, 3]
```

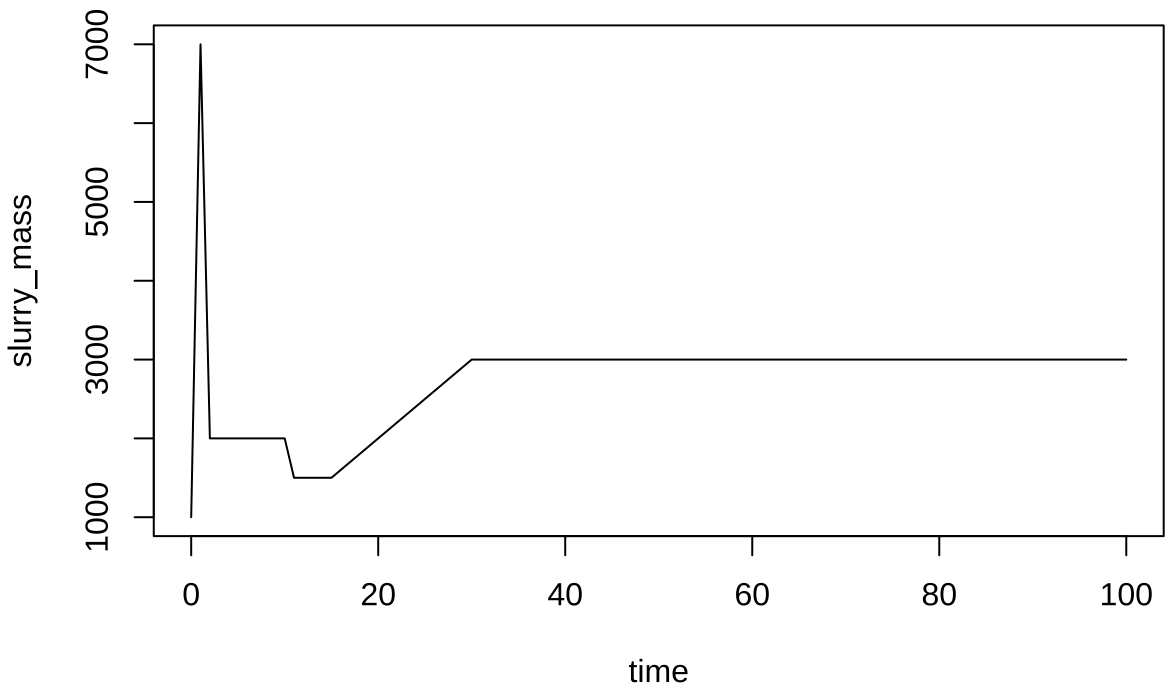
```
## [[1]]
## VSd
##   0
```

```r
devtools::load_all()
```

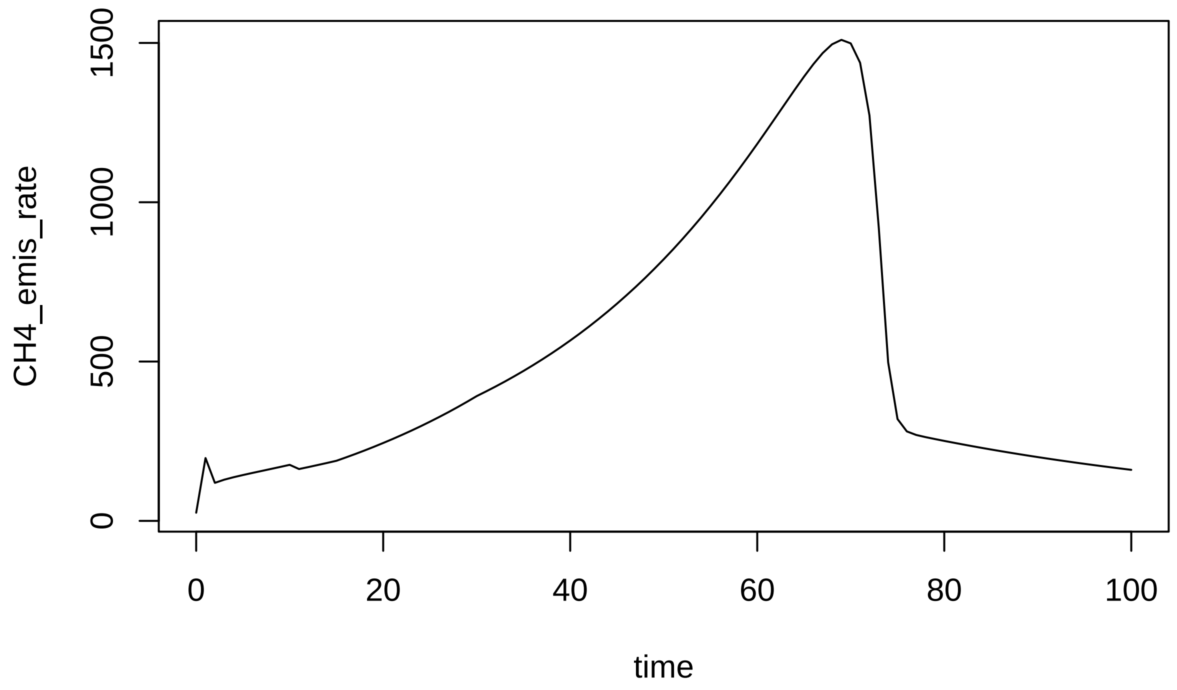
```
## i Loading ABM
```

```r
out4a <- abm(100,
             mng_pars = mng_pars,
             man_pars = man_pars,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars,
             var_pars = var_pars)
```

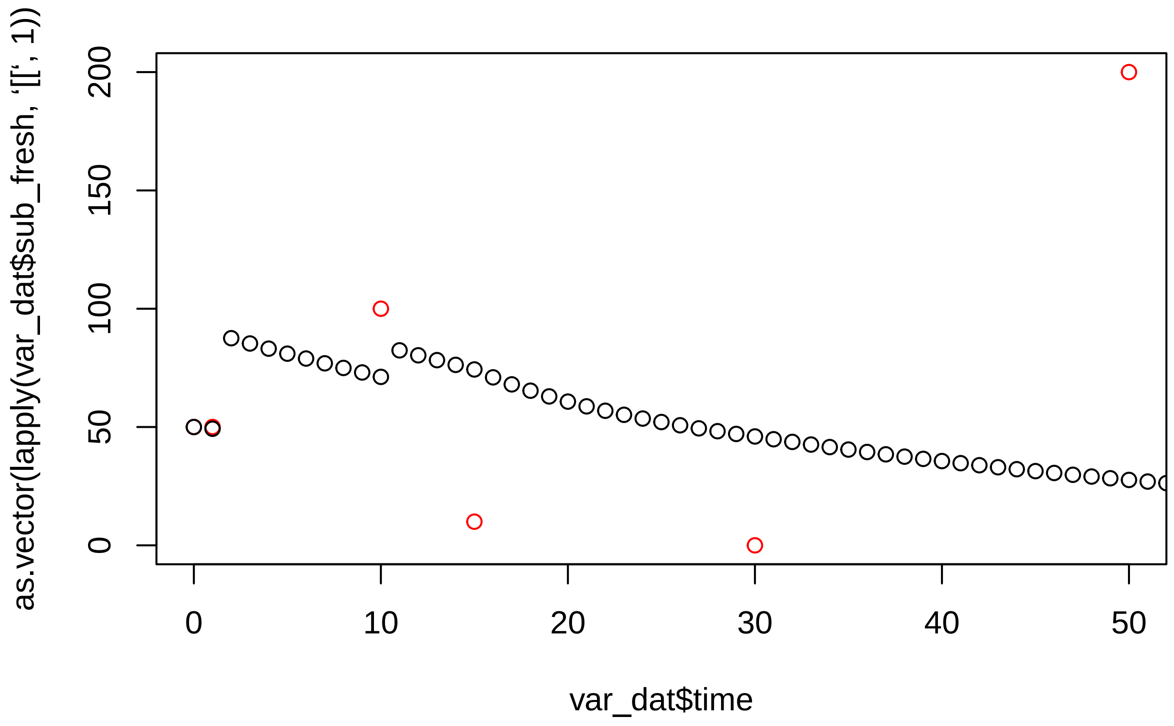
```r
plot(slurry_mass ~ time, data = out4a, type = 'l')
```



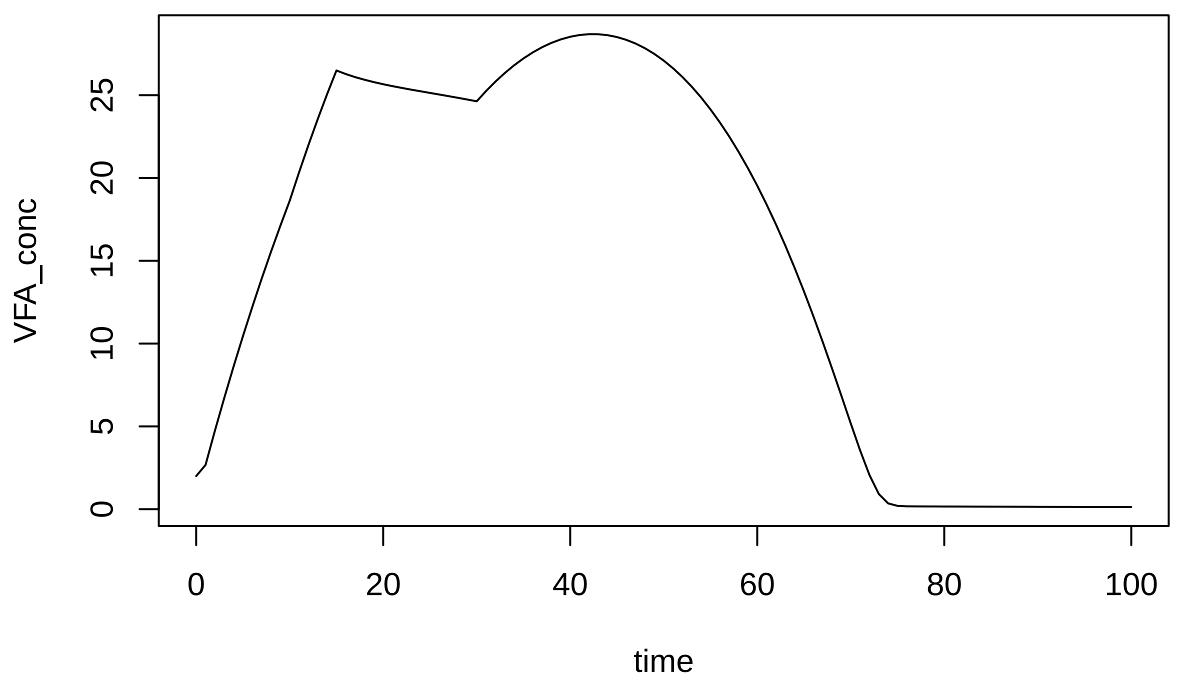
```r
plot(CH4_emis_rate ~ time, data = out4a, type = 'l')
```



```r
plot(var_dat$time, as.vector(lapply(var_dat$sub_fresh, `[[`, 1)), type = 'p', col = 'red')
lines(VSd_conc ~ time, data = out4a, type = 'p')
```



```r
plot(VFA_conc ~ time, data = out4a, type = 'l')
```



Let's vary some microbial parameters as well. And pH.

```
var_dat <- data.frame(time = c(0, 1, 10, 15, 30, 50),
                      slurry_mass = c(1000, 7000, 2000, 5000, 3000, 10000),
                      pH = 7 - 0:5/10)
var_dat
```

```
##   time slurry_mass  pH
## 1    0        1000 7.0
## 2    1        7000 6.9
## 3   10        2000 6.8
## 4   15        5000 6.7
## 5   30        3000 6.6
## 6   50       10000 6.5
```

VSd.

```
var_dat$sub_fresh <- rep(list(c(VSd = 50)), nrow(var_dat))
var_dat$sub_fresh[3] <- list(c(VSd = 100))
```

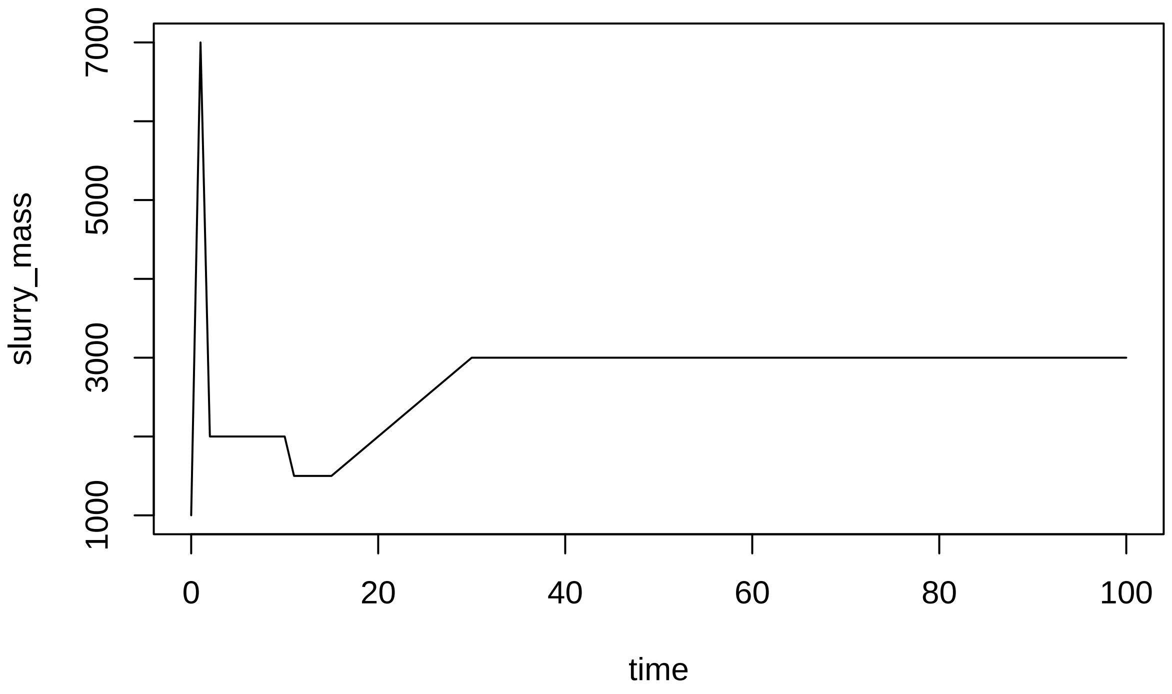
Some microbial parameters for a shift in temperature optima, "adaptation" for example.

```
for (i in 1:nrow(var_dat)) {
  var_dat$T_opt[i] <- list(grp_pars$T_opt + 2 * i)
}
var_dat
```

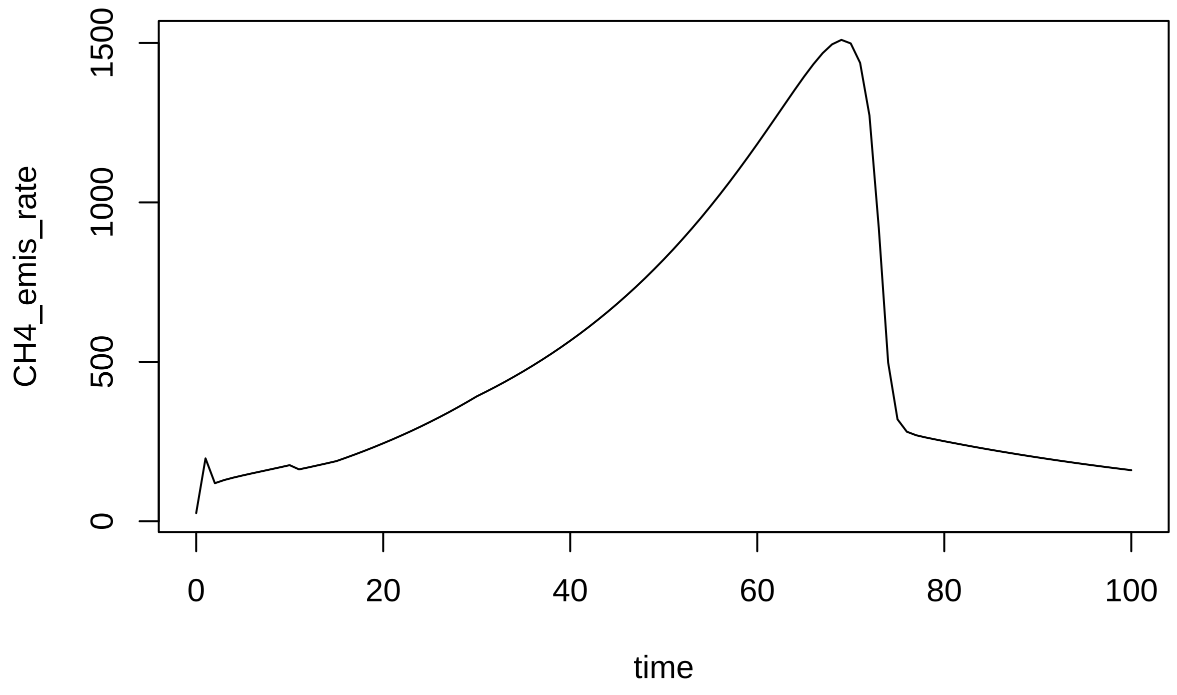
```
##   time slurry_mass  pH sub_fresh          T_opt
## 1    0        1000 7.0        50 20, 20, 30, 46
## 2    1        7000 6.9        50 22, 22, 32, 48
## 3   10        2000 6.8       100 24, 24, 34, 50
## 4   15        5000 6.7        50 26, 26, 36, 52
## 5   30        3000 6.6        50 28, 28, 38, 54
## 6   50       10000 6.5        50 30, 30, 40, 56
```

```r
out4b <- abm(100,
             mng_pars = mng_pars,
             man_pars = man_pars,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars,
             var_pars = var_pars)
```

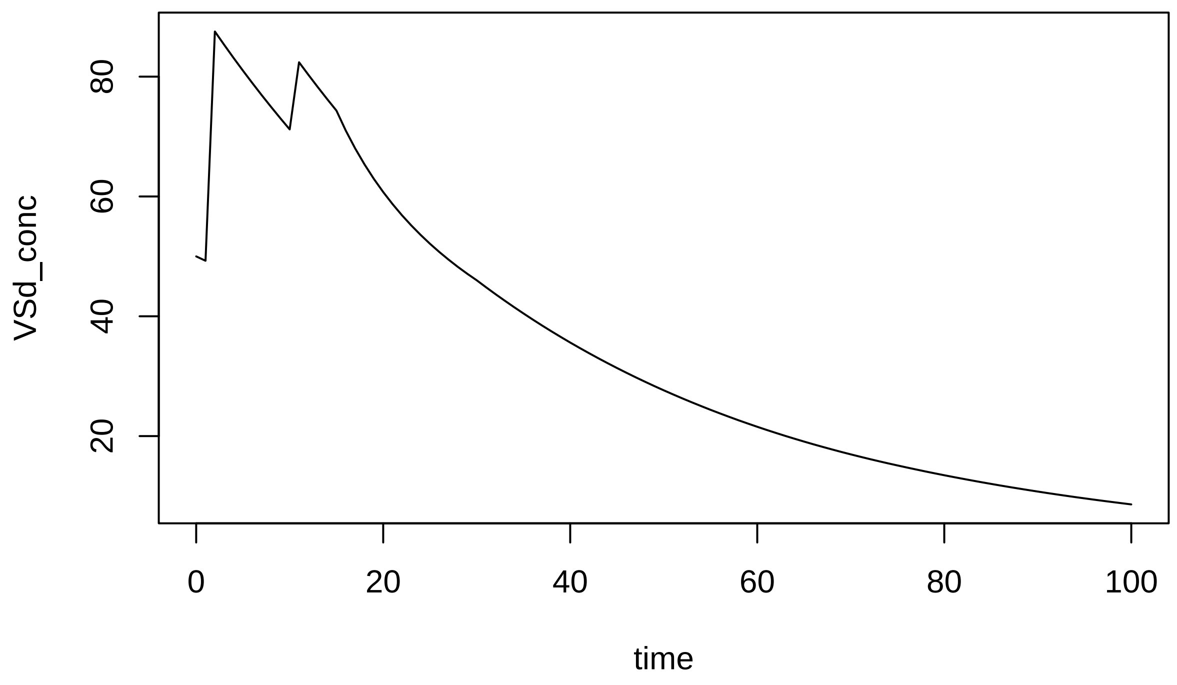
```r
plot(slurry_mass ~ time, data = out4b, type = 'l')
```



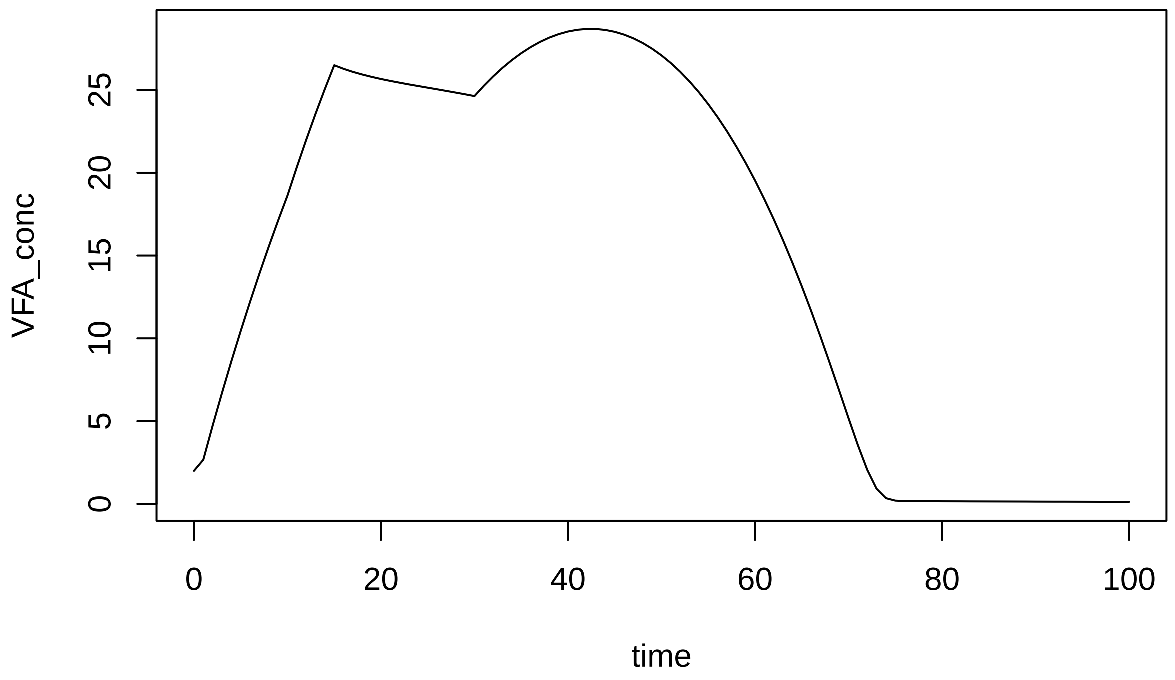
```r
plot(CH4_emis_rate ~ time, data = out4b, type = 'l')
```



```r
plot(VSd_conc ~ time, data = out4b, type = 'l')
```



```r
plot(VFA_conc ~ time, data = out4b, type = 'l')
```



5. Time-variable inputs part 3

The list-in-data frame approach is clunky. Here is an alternative.

```
var_dat <- data.frame(time = c(0, 1, 10, 15, 30, 50),
                      slurry_mass = c(1000, 7000, 2000, 5000, 3000, 10000))
var_dat
```

```
##   time slurry_mass
## 1    0        1000
## 2    1        7000
## 3   10        2000
## 4   15        5000
## 5   30        3000
## 6   50       10000
```

Make a separate data frame for each other argument (any name, but note column names!).

```
sub_fresh_dat = data.frame(time = c(0, 1, 10, 15, 30, 50),
                      VSd = c(50, 100, 0, 0, 200, 50))
```

```
T_opt_dat = data.frame(time = c(0, 1, 10, 15, 30, 50),
                      m1 = 20 + 0:5 * 2,
                      m2 = 20 + 0:5 * 2,
                      m3 = 30 + 0:5 * 2,
                      sr1 = 46 + 0:5 * 2)
```

and combine them in a list, using the parameter element names for element names (e.g., `sub_fresh` is the name of an element in `sub_pars`).

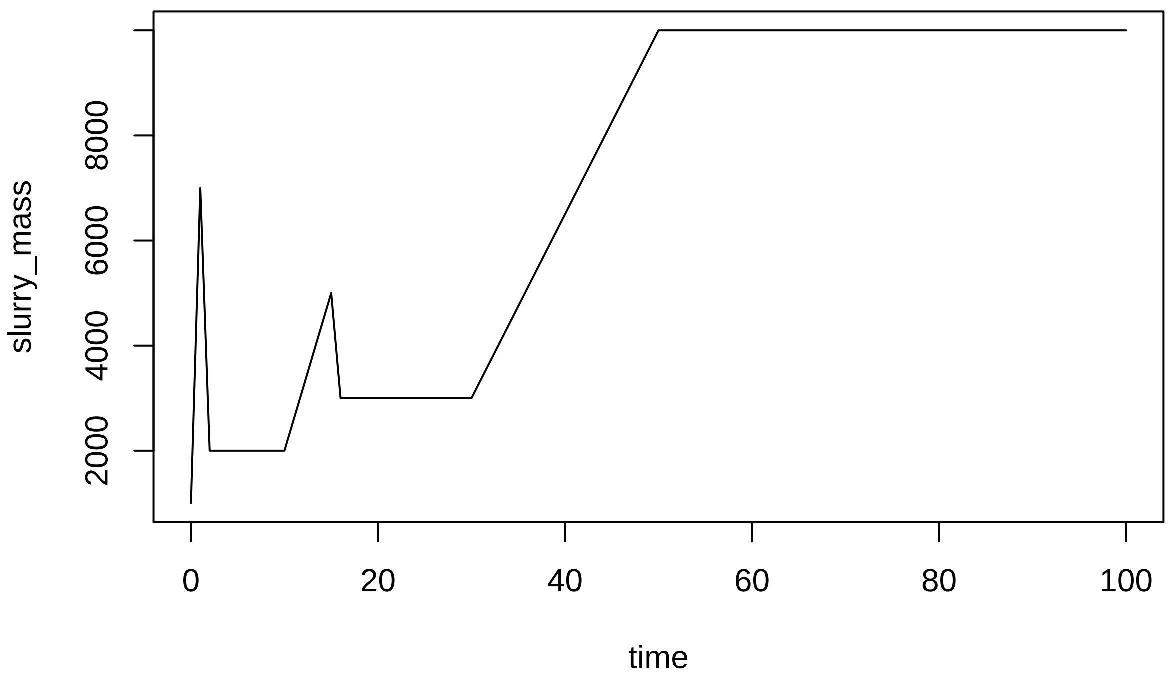
```
var_pars <- list(var = var_dat, sub_fresh = sub_fresh_dat, T_opt = T_opt_dat)
```

```
devtools::load_all()
```

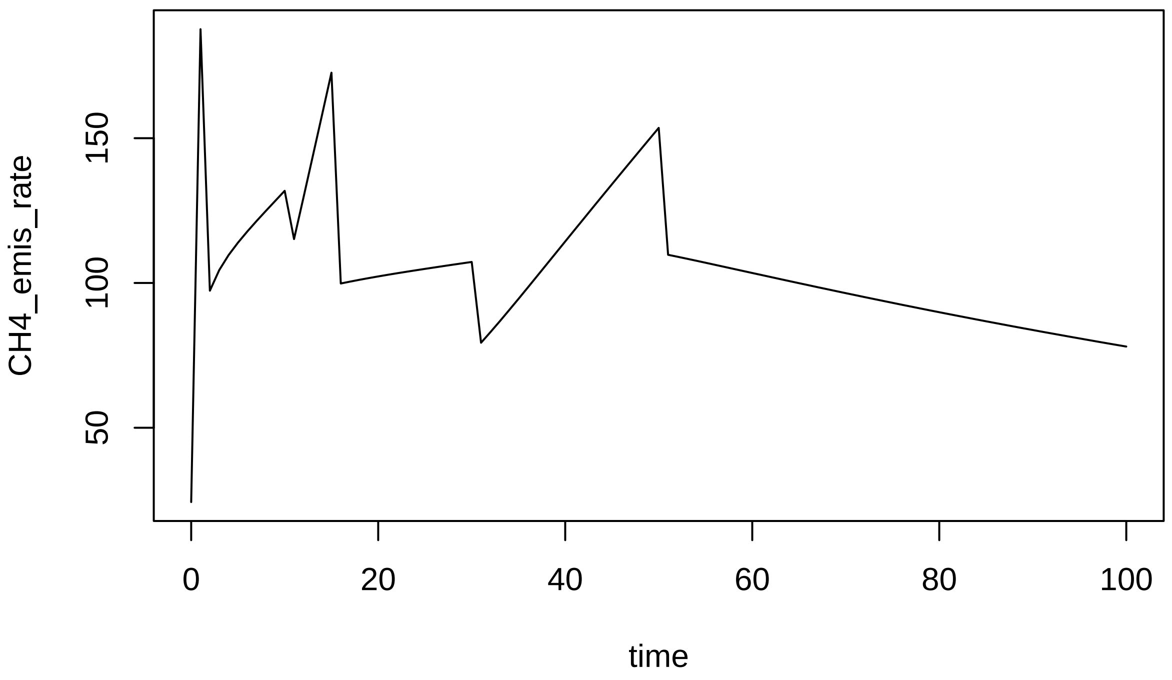
```
## i Loading ABM
```

```r
out5 <- abm(100,
             mng_pars = mng_pars,
             man_pars = man_pars,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars,
             var_pars = var_pars)
```

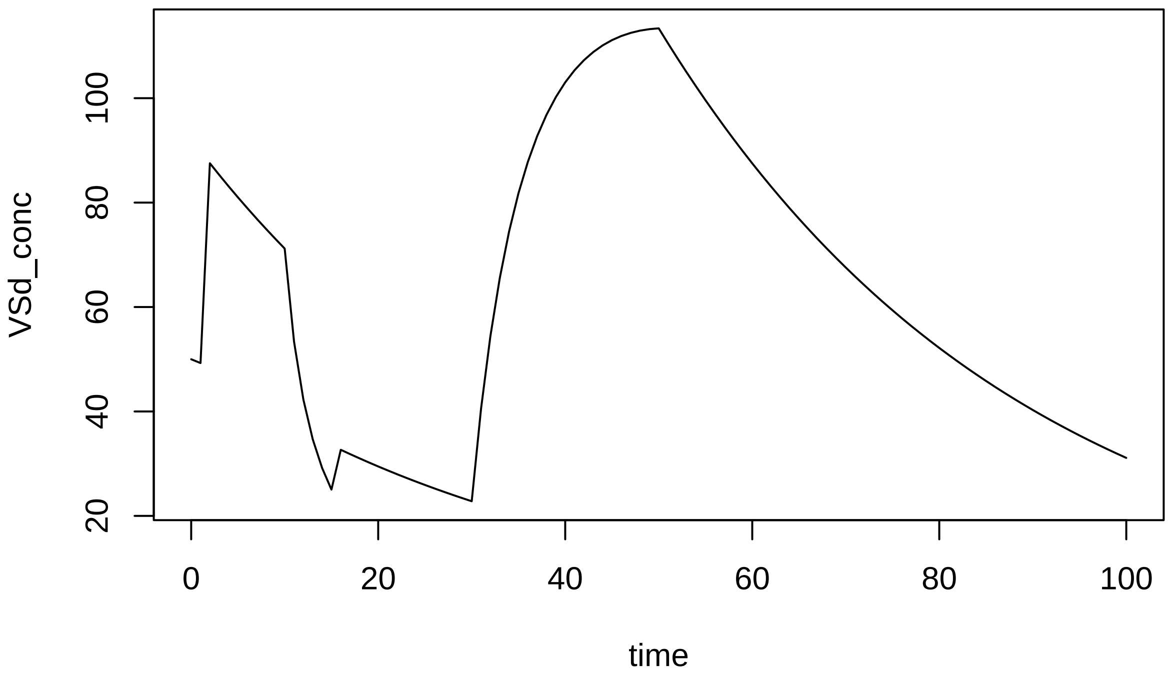
```r
plot(slurry_mass ~ time, data = out5, type = 'l')
```



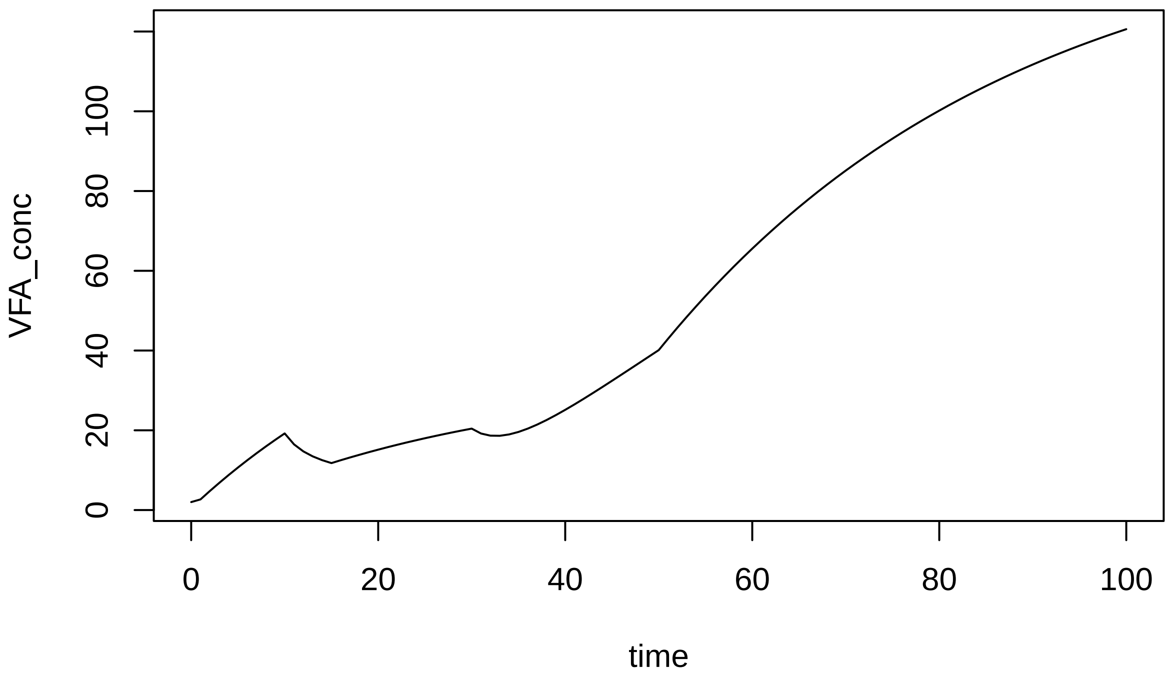
```r
plot(CH4_emis_rate ~ time, data = out5, type = 'l')
```



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



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



6. Flexible solutes

Any conservative solute can be added in `man_pars`, using any names. I am moving toward using a "master species" approach, so it makes sense to use the chemical formula of the primary species, with `p` or `m` for a charge symbol. The `comp` part of the name below is for "component".

```
man_pars6 <- list(comps = c('H2S', 'SO4m2', 'NH4p'),
                  comp_fresh = c(H2S = 0.01, SO4m2 = 0.2, NH4p = 2.5),
                  VFA_fresh = c(VFA = 2),
                  pH = 7, dens = 1000)
```

Note that `VFA` is still special–it has a fixed name in the code and is not conservative.

```
devtools::load_all()
```

```
## i Loading ABM
```

```
out6a <- abm(365,
             mng_pars = mng_pars,
             man_pars = man_pars6,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars)
```

```
## Warning in checkCOD(dat = dat, grps = pars$grps, subs = pars$subs, COD_conv =
## pars$COD_conv, : COD balance is off by 1.7%
```

```
tail(out6a)
```

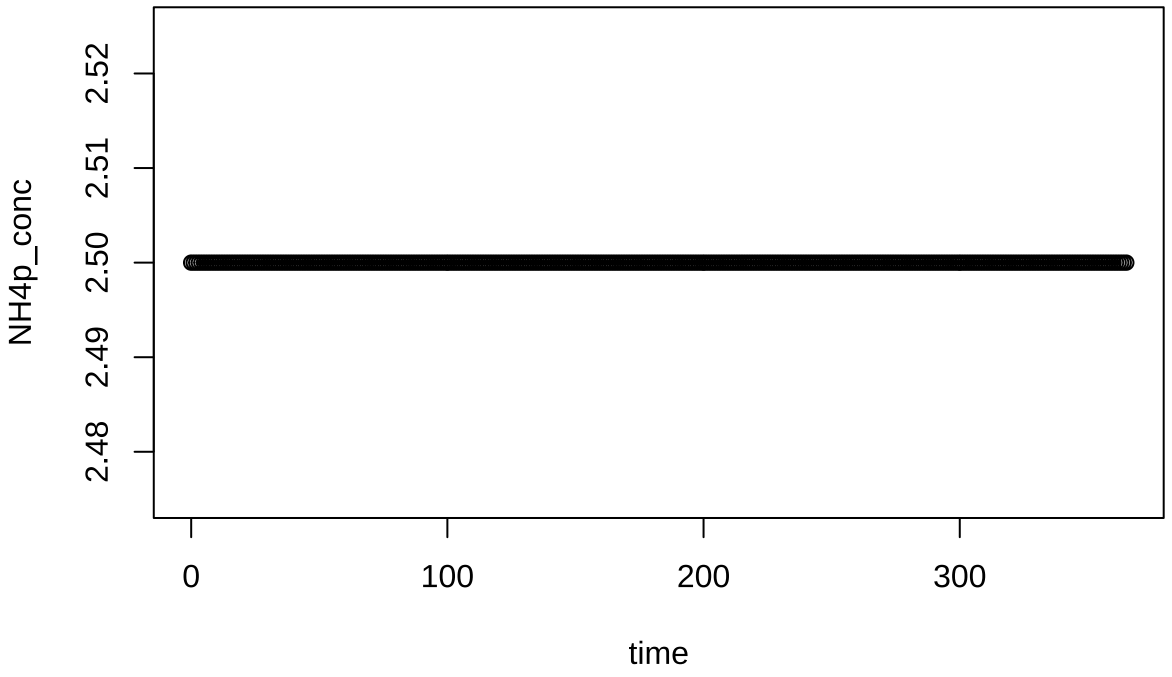
```
##     time       m0       m1       m2      sr1      VSd  H2S  SO4m2    NH4p
## 364  360 74909.98 72497.35 337403.6 17643.06 15468816 6100 122000 1525000
## 365  361 77056.84 74543.26 351549.2 17788.74 15576604 6200 124000 1550000
## 366  362 79234.75 76617.42 366160.3 17931.53 15682002 6300 126000 1575000
## 367  363 81441.36 78717.58 381233.0 18071.50 15785084 6400 128000 1600000
## 368  364 83673.78 80840.93 396758.5 18208.69 15885918 6500 130000 1625000
## 369  365 85928.37 82984.01 412722.1 18343.17 15984571 6600 132000 1650000
##         VFA slurry_mass CH4_emis_cum CO2_emis_cum slurry_load  COD_load
## 364 4796289      610000     26846605     21417315     3600000 187920000
## 365 4684393      620000     26973866     21518840     3610000 188442000
## 366 4557322      630000     27105400     21623774     3620000 188964000
## 367 4414981      640000     27241214     21732122     3630000 189486000
## 368 4257395      650000     27381289     21843868     3640000 190008000
## 369 4084743      660000     27525568     21958969     3650000 190530000
##     CH4_emis_rate temp_C pH   m0_eff   m1_eff  m2_eff  sr1_eff  VSd_eff H2S_eff
## 364      125134.1     20  7 441740.7 422210.1 2239891 63286.93 53531054   29910
## 365      129394.2     20  7 441740.7 422210.1 2239891 63286.93 53531054   29910
## 366      133674.8     20  7 441740.7 422210.1 2239891 63286.93 53531054   29910
## 367      137950.5     20  7 441740.7 422210.1 2239891 63286.93 53531054   29910
## 368      142189.3     20  7 441740.7 422210.1 2239891 63286.93 53531054   29910
## 369      146351.3     20  7 441740.7 422210.1 2239891 63286.93 53531054   29910
##     SO4m2_eff NH4p_eff VFA_eff slurry_mass_eff slurry_depth   m0_conc   m1_conc
## 364    598200  7477500 3419880         2991000          6.1 0.1228032 0.1188481
## 365    598200  7477500 3419880         2991000          6.2 0.1242852 0.1202311
## 366    598200  7477500 3419880         2991000          6.3 0.1257694 0.1216150
## 367    598200  7477500 3419880         2991000          6.4 0.1272521 0.1229962
## 368    598200  7477500 3419880         2991000          6.5 0.1287289 0.1243707
## 369    598200  7477500 3419880         2991000          6.6 0.1301945 0.1257334
##       m2_conc   sr1_conc VSd_conc H2S_conc SO4m2_conc NH4p_conc VFA_conc
## 364 0.5531206 0.02892305 25.35872     0.01        0.2       2.5 7.862769
## 365 0.5670148 0.02869152 25.12355     0.01        0.2       2.5 7.555472
## 366 0.5812069 0.02846275 24.89207     0.01        0.2       2.5 7.233845
## 367 0.5956766 0.02823671 24.66419     0.01        0.2       2.5 6.898408
## 368 0.6103976 0.02801337 24.43987     0.01        0.2       2.5 6.549838
## 369 0.6253365 0.02779268 24.21905     0.01        0.2       2.5 6.189004
##     m0_eff_conc m1_eff_conc m2_eff_conc sr1_eff_conc VSd_eff_conc H2S_eff_conc
## 364     0.14769   0.1411602   0.7488771   0.02115912     17.89738         0.01
## 365     0.14769   0.1411602   0.7488771   0.02115912     17.89738         0.01
## 366     0.14769   0.1411602   0.7488771   0.02115912     17.89738         0.01
## 367     0.14769   0.1411602   0.7488771   0.02115912     17.89738         0.01
## 368     0.14769   0.1411602   0.7488771   0.02115912     17.89738         0.01
## 369     0.14769   0.1411602   0.7488771   0.02115912     17.89738         0.01
##     SO4m2_eff_conc NH4p_eff_conc VFA_eff_conc
## 364            0.2           2.5      1.14339
## 365            0.2           2.5      1.14339
## 366            0.2           2.5      1.14339
## 367            0.2           2.5      1.14339
## 368            0.2           2.5      1.14339
## 369            0.2           2.5      1.14339
```

```r
head(out6a)
```

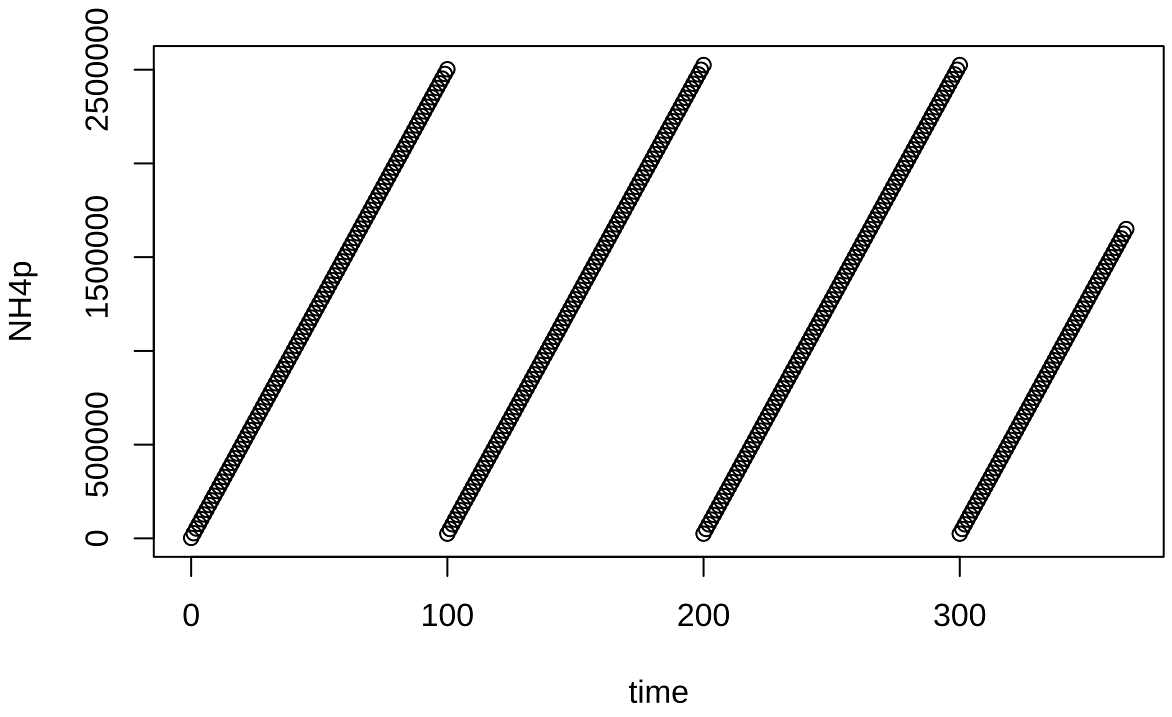
```
##   time        m0        m1        m2       sr1       VSd H2S SO4m2   NH4p
## 1    0   50.0000   50.0000   50.0000   50.0000   50000.0  10   200   2500
## 2    1  554.0098  553.8533  558.3748  544.0431  542318.4 110  2200  27500
## 3    2 1066.2767 1065.6732 1083.1940 1028.3035 1022076.9 210  4200  52500
## 4    3 1588.3161 1586.9430 1627.0197 1502.9749 1489597.5 310  6200  77500
## 5    4 2121.2034 2118.7114 2191.8594 1968.2472 1945194.0 410  8200 102500
## 6    5 2665.7726 2661.7877 2779.4361 2424.3064 2389172.2 510 10200 127500
##         VFA slurry_mass CH4_emis_cum CO2_emis_cum slurry_load COD_load
## 1   2000.00        1000       0.0000       0.0000          0        0
## 2  29018.66       11000     163.6272     130.5362      10000   522000
## 3  67371.44       21000     628.8119     501.6449      20000  1044000
## 4 116611.44       31000    1425.4337    1137.1629      30000  1566000
## 5 176326.70       41000    2577.0208    2055.8603      40000  2088000
## 6 246127.08       51000    4103.8067    3273.8785      50000  2610000
##   CH4_emis_rate temp_C pH m0_eff m1_eff m2_eff sr1_eff VSd_eff H2S_eff
## 1      25.52844     20  7      0      0      0       0       0       0
## 2     308.65512     20  7      0      0      0       0       0       0
## 3     626.59326     20  7      0      0      0       0       0       0
## 4     970.52564     20  7      0      0      0       0       0       0
## 5    1335.99741     20  7      0      0      0       0       0       0
## 6    1720.64113     20  7      0      0      0       0       0       0
##   SO4m2_eff NH4p_eff VFA_eff slurry_mass_eff slurry_depth    m0_conc    m1_conc
## 1         0        0       0               0         0.01 0.05000000 0.05000000
## 2         0        0       0               0         0.11 0.05036453 0.05035030
## 3         0        0       0               0         0.21 0.05077508 0.05074634
## 4         0        0       0               0         0.31 0.05123600 0.05119171
## 5         0        0       0               0         0.41 0.05173667 0.05167589
## 6         0        0       0               0         0.51 0.05227005 0.05219192
##      m2_conc   sr1_conc VSd_conc H2S_conc SO4m2_conc NH4p_conc VFA_conc
## 1 0.05000000 0.05000000 50.00000     0.01        0.2       2.5 2.000000
## 2 0.05076135 0.04945846 49.30167     0.01        0.2       2.5 2.638060
## 3 0.05158067 0.04896683 48.67033     0.01        0.2       2.5 3.208164
## 4 0.05248451 0.04848306 48.05153     0.01        0.2       2.5 3.761660
## 5 0.05345999 0.04800603 47.44376     0.01        0.2       2.5 4.300651
## 6 0.05449875 0.04753542 46.84651     0.01        0.2       2.5 4.826021
##   m0_eff_conc m1_eff_conc m2_eff_conc sr1_eff_conc VSd_eff_conc H2S_eff_conc
## 1         NaN         NaN         NaN          NaN          NaN          NaN
## 2         NaN         NaN         NaN          NaN          NaN          NaN
## 3         NaN         NaN         NaN          NaN          NaN          NaN
## 4         NaN         NaN         NaN          NaN          NaN          NaN
## 5         NaN         NaN         NaN          NaN          NaN          NaN
## 6         NaN         NaN         NaN          NaN          NaN          NaN
##   SO4m2_eff_conc NH4p_eff_conc VFA_eff_conc
## 1            NaN           NaN          NaN
## 2            NaN           NaN          NaN
## 3            NaN           NaN          NaN
## 4            NaN           NaN          NaN
## 5            NaN           NaN          NaN
## 6            NaN           NaN          NaN
```

Conservative components are boring in output (without inhibition or volatilization).

```r
plot(NH4p_conc ~ time, data = out6a)
```



```
plot(NH4p ~ time, data = out6a)
```



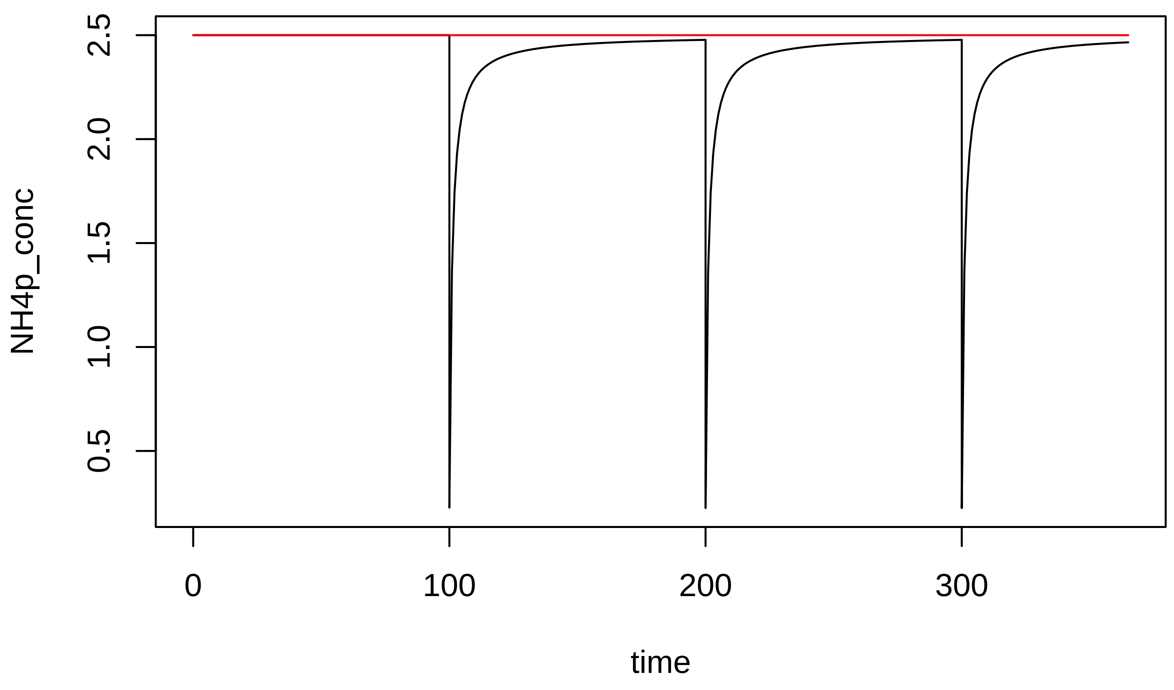
We can see dilution effects at least if some washing water is added.

```
mng_pars6 = list(slurry_prod_rate = 10000,
                 slurry_mass = 1000,
                 storage_depth = 2,
                 resid_depth = 0.1,
                 area = 100,
                 empty_int = 100,
                 temp_C = 20,
                 wash_water = 100000,
                 wash_int = 100,
                 rest_d = 0,
                 resid_enrich = 1)
```

```
out6b <- abm(365,
             mng_pars = mng_pars6,
             man_pars = man_pars6,
             grp_pars = grp_pars,
             mic_pars = mic_pars,
             sub_pars = sub_pars,
             chem_pars = chem_pars)
```

```
## Warning in checkCOD(dat = dat, grps = pars$grps, subs = pars$subs, COD_conv =
## pars$COD_conv, : COD balance is off by 32%
```

```
plot(NH4p_conc ~ time, data = out6b, type = 'l')
lines(NH4p_conc ~ time, data = out6a, col = 'red')
```



7. Speciation

Acid-base reactions are needed for inhibition. They can be added for any component. The `chem_pars` argument can accept temperature-dependent log ka expressions.

```
man_pars7 <- list(comps = c('H2S', 'NH4p'),
                  comp_fresh = c(H2S = 0.01, NH4p = 2.5),
                  VFA_fresh = c(VFA = 2),
                  pH = 7, dens = 1000)

chem_pars7 <- list(COD_conv = c(CH4 = 1/0.2507, xa = 1/0.7069561,
                                VFA = 1/0.9383125, S = 1/0.5015, VS = 1/0.69,
                                CO2_aer = 1/0.436, CO2_sr = 1/1.2,
                                C_xa = 1/0.3753125),
                    specs = c('NH3', 'HSm', 'VFAm'),
                    mspec = c(NH3 = 'NH4p', HSm = 'H2S', VFAm = 'VFA'),
                    lka = c(NH3 = '- 0.09046 - 2729.31/temp_K',
                            HSm = '- 3448.7/temp_K + 47.479 - 7.5227 * log(temp_K)',
                            VFAm = '-4.8288 + 21.42/temp_K')
)
```

```
out7 <- abm(365,
            mng_pars = mng_pars,
            man_pars = man_pars6,
            grp_pars = grp_pars,
            mic_pars = mic_pars,
            sub_pars = sub_pars,
            chem_pars = chem_pars7)
```

```
## Warning in checkCOD(dat = dat, grps = pars$grps, subs = pars$subs, COD_conv =
## pars$COD_conv, : COD balance is off by 1.7%
```

```
head(out7)
```

```
##   time        m0        m1        m2       sr1       VSd H2S SO4m2   NH4p
## 1    0   50.0000   50.0000   50.0000   50.0000   50000.0  10   200   2500
## 2    1  554.0098  553.8533  558.3748  544.0431  542318.4 110  2200  27500
## 3    2 1066.2767 1065.6732 1083.1940 1028.3035 1022076.9 210  4200  52500
## 4    3 1588.3161 1586.9430 1627.0197 1502.9749 1489597.5 310  6200  77500
## 5    4 2121.2034 2118.7114 2191.8594 1968.2472 1945194.0 410  8200 102500
## 6    5 2665.7726 2661.7877 2779.4361 2424.3064 2389172.2 510 10200 127500
##         VFA slurry_mass CH4_emis_cum CO2_emis_cum slurry_load COD_load
## 1   2000.00        1000       0.0000       0.0000           0        0
## 2  29018.66       11000     163.6272     130.5362       10000   522000
## 3  67371.44       21000     628.8119     501.6449       20000  1044000
## 4 116611.44       31000    1425.4337    1137.1629       30000  1566000
## 5 176326.70       41000    2577.0208    2055.8603       40000  2088000
## 6 246127.08       51000    4103.8067    3273.8785       50000  2610000
##   CH4_emis_rate temp_C pH m0_eff m1_eff m2_eff sr1_eff VSd_eff H2S_eff
## 1      25.52844     20  7      0      0      0       0       0       0
## 2     308.65512     20  7      0      0      0       0       0       0
## 3     626.59326     20  7      0      0      0       0       0       0
## 4     970.52564     20  7      0      0      0       0       0       0
## 5    1335.99741     20  7      0      0      0       0       0       0
## 6    1720.64113     20  7      0      0      0       0       0       0
##   SO4m2_eff NH4p_eff VFA_eff slurry_mass_eff slurry_depth    m0_conc    m1_conc
## 1         0        0       0               0         0.01 0.05000000 0.05000000
## 2         0        0       0               0         0.11 0.05036453 0.05035030
## 3         0        0       0               0         0.21 0.05077508 0.05074634
## 4         0        0       0               0         0.31 0.05123600 0.05119171
## 5         0        0       0               0         0.41 0.05173667 0.05167589
## 6         0        0       0               0         0.51 0.05227005 0.05219192
##      m2_conc   sr1_conc VSd_conc H2S_conc SO4m2_conc NH4p_conc VFA_conc
## 1 0.05000000 0.05000000 50.00000     0.01        0.2       2.5 2.000000
## 2 0.05076135 0.04945846 49.30167     0.01        0.2       2.5 2.638060
## 3 0.05158067 0.04896683 48.67033     0.01        0.2       2.5 3.208164
## 4 0.05248451 0.04848306 48.05153     0.01        0.2       2.5 3.761660
## 5 0.05345999 0.04800603 47.44376     0.01        0.2       2.5 4.300651
## 6 0.05449875 0.04753542 46.84651     0.01        0.2       2.5 4.826021
##   m0_eff_conc m1_eff_conc m2_eff_conc sr1_eff_conc VSd_eff_conc H2S_eff_conc
## 1         NaN         NaN         NaN          NaN          NaN          NaN
## 2         NaN         NaN         NaN          NaN          NaN          NaN
## 3         NaN         NaN         NaN          NaN          NaN          NaN
## 4         NaN         NaN         NaN          NaN          NaN          NaN
## 5         NaN         NaN         NaN          NaN          NaN          NaN
## 6         NaN         NaN         NaN          NaN          NaN          NaN
##   SO4m2_eff_conc NH4p_eff_conc VFA_eff_conc
## 1            NaN           NaN          NaN
## 2            NaN           NaN          NaN
## 3            NaN           NaN          NaN
## 4            NaN           NaN          NaN
## 5            NaN           NaN          NaN
## 6            NaN           NaN          NaN
```

But chemical species are actually ignored unless they are used in inhibition.

8. Inhibition

Any chemical species can inhibit any microbial group. Inhibition parameters (currently initial and complete concentrations, with linear response, why not?) are entered in a matrix.

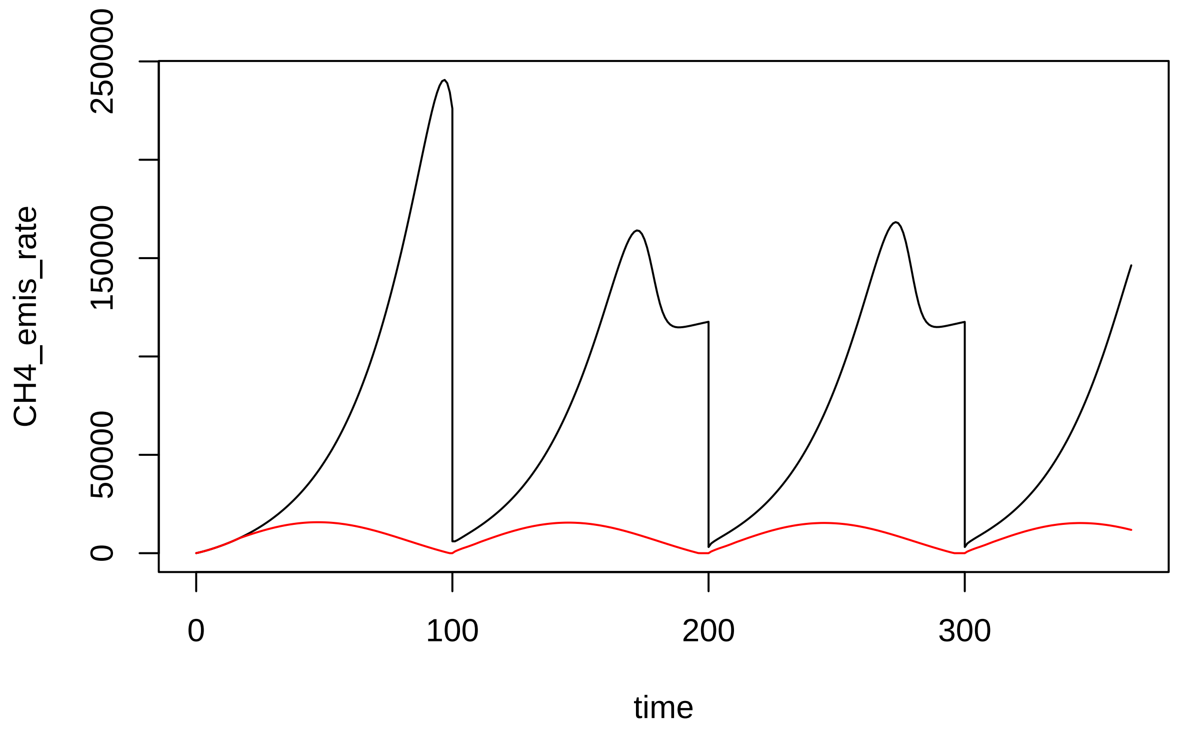
```
ilwr <- matrix(
  c(5, 0.2, 10, 0.5,
    5, 0.2, 10, 0.5,
    5, 0.2, 10, 0.5,
    5, 0.2, 10, 0.5),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(
    c('m0', 'm1', 'm2', 'sr1'),
    c('NH4p', 'NH3', 'VFAm', 'VFA')
  )
)

iupr <- matrix(
  c(9, 0.9, 30, 1,
    9, 0.9, 30, 1,
    9, 0.9, 30, 1,
    9, 0.9, 30, 1),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(
    c('m0', 'm1', 'm2', 'sr1'),
    c('NH4p', 'NH3', 'VFAm', 'VFA')
  )
)

inhib_pars <- list(
  ilwr = ilwr,
  iupr = iupr
)

inhib_pars
```

```
## $ilwr
##     NH4p NH3 VFAm VFA
## m0     5 0.2   10 0.5
## m1     5 0.2   10 0.5
## m2     5 0.2   10 0.5
## sr1    5 0.2   10 0.5
##
## $iupr
##     NH4p NH3 VFAm VFA
## m0     9 0.9   30   1
## m1     9 0.9   30   1
## m2     9 0.9   30   1
## sr1    9 0.9   30   1
```

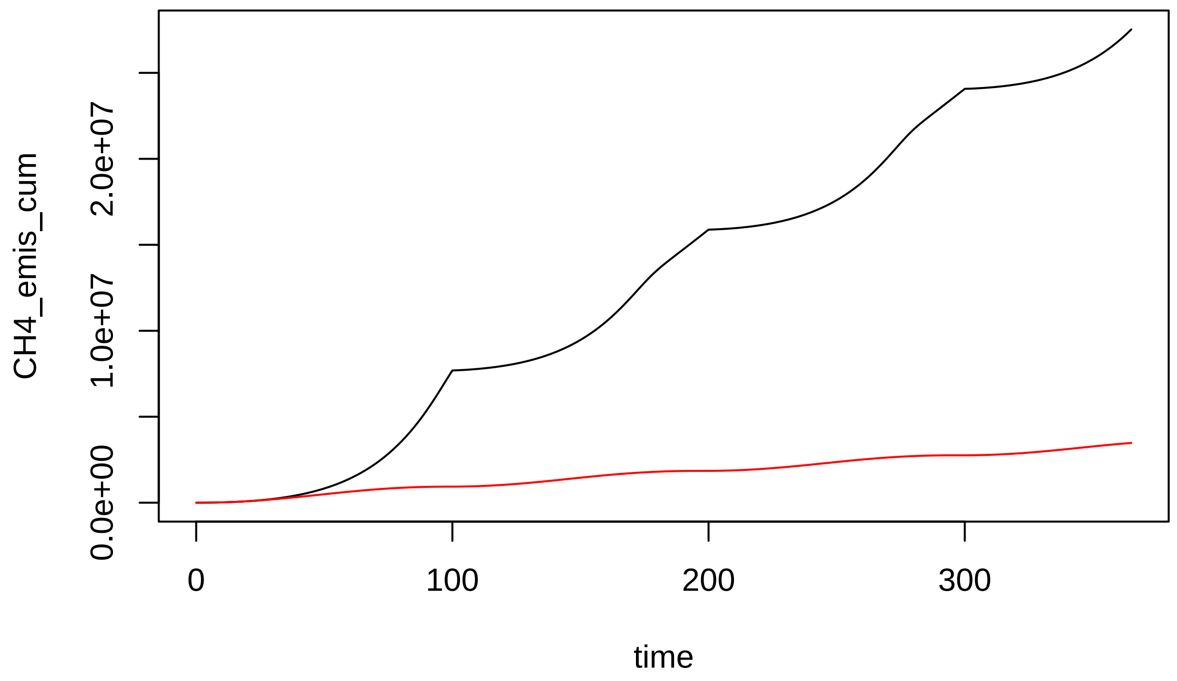
```
man_pars8 <- list(comps = c('H2S', 'NH4p'),
                 comp_fresh = c(H2S = 0.01,
                                      NH4p = 2.5),
                 VFA_fresh = c(VFA = 2),
                 pH = 7, dens = 1000)

chem_pars8 <- list(COD_conv = c(CH4 = 1/0.2507, xa = 1/0.7069561,
                               VFA = 1/0.9383125, S = 1/0.5015, VS = 1/0.69,
                               CO2_aer = 1/0.436, CO2_sr = 1/1.2,
                               C_xa = 1/0.3753125),
                   specs = c('NH3', 'HSm', 'VFAm'),
                   mspec = c(NH3 = 'NH4p', HSm = 'H2S', VFAm = 'VFA'),
                   lka = c(NH3 = '- 0.09046 - 2729.31/temp_K',
                           HSm = '- 3448.7/temp_K + 47.479 - 7.5227* log(temp_K)',
                           VFAm = '-4.8288 + 21.42/temp_K')
)
```

```
out8 <- abm(365,
            mng_pars = mng_pars,
            man_pars = man_pars7,
            grp_pars = grp_pars,
            mic_pars = mic_pars,
            sub_pars = sub_pars,
            chem_pars = chem_pars7,
            inhib_pars = inhib_pars
)
```

```
plot(CH4_emis_rate ~ time, data = out7, type = 'l')
lines(CH4_emis_rate ~ time, data = out8, col = 'red')
```



```
plot(CH4_emis_cum ~ time, data = out7, type = 'l')
lines(CH4_emis_cum ~ time, data = out8, col = 'red')
```



9. COD balance

There is now a `checkCOD()` function that runs on `abm()` results before returning them. For now the tolerance is fixed at 1%. Some of the examples above do not meet that criterion for some reason. At least one shows a real problem that needs to be identified.