

Demo of new simple1 version

Sasha D. Hafner

30 May, 2025 11:37

Overview

This demo shows:

1. basic usage,
2. variable substrates,
3. time-variable parameters
4. COD balance,

Prep

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

```
## i Loading ABM
```

Function arguments

The argument list currently looks like this:

```
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 = man_pars$conc_fresh),  
  grp_pars,  
  mic_pars,  
  sub_pars,  
  chem_pars,  
  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  
  starting = NULL,                           # Output from previous simulation to be starting condition  
  value = 'ts',                              # Type of output  
  warn = TRUE) {
```

The main changes are:

- `sub_pars` for defining substrates
- `ctrl_pars` for some “control” parameters
- `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.

```
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 now (like anything here, it could change).

```
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 there is no inhibition for now.

```
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”.

```
mic_pars <- list(ks_S04 = 0.00694,
                km_urea = 0.913,
                dd_rate_xa = 0.02)
```

These last two are similar to other versions.

```

man_pars <- list(conc_fresh = c(sulfide = 0.01,
                                sulfate = 0.2,
                                TAN = 0.0,
                                VFA = 2,
                                ash = 15),
                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
```

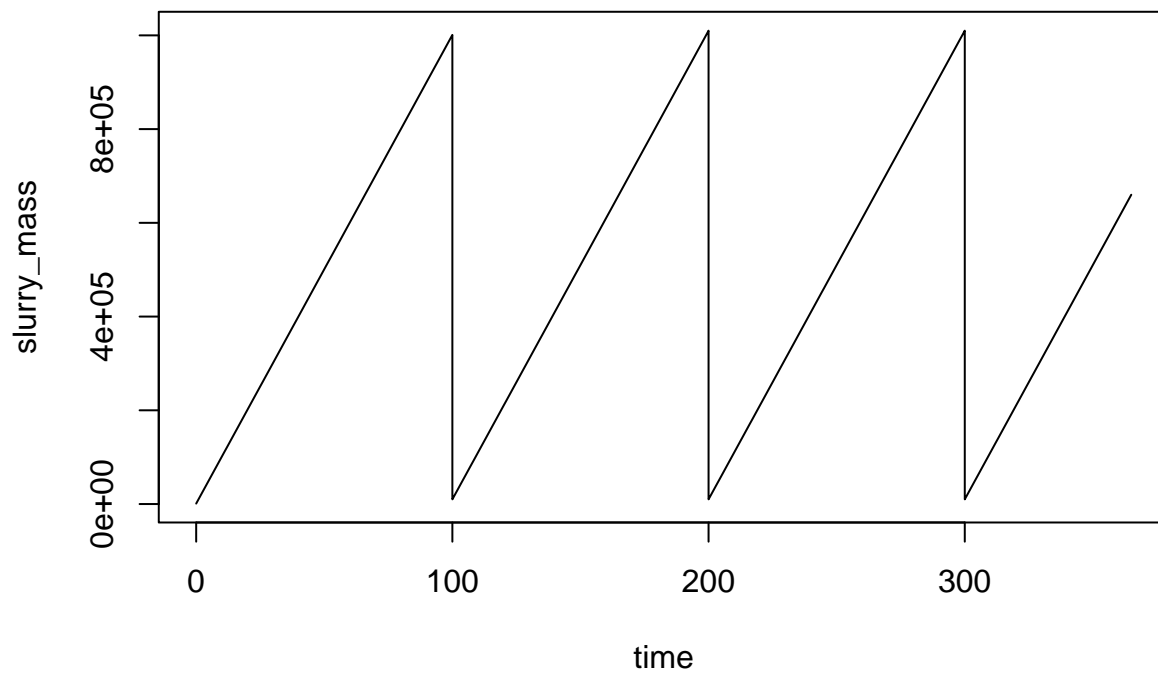
```
tail(out1)
```

```
##      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
```

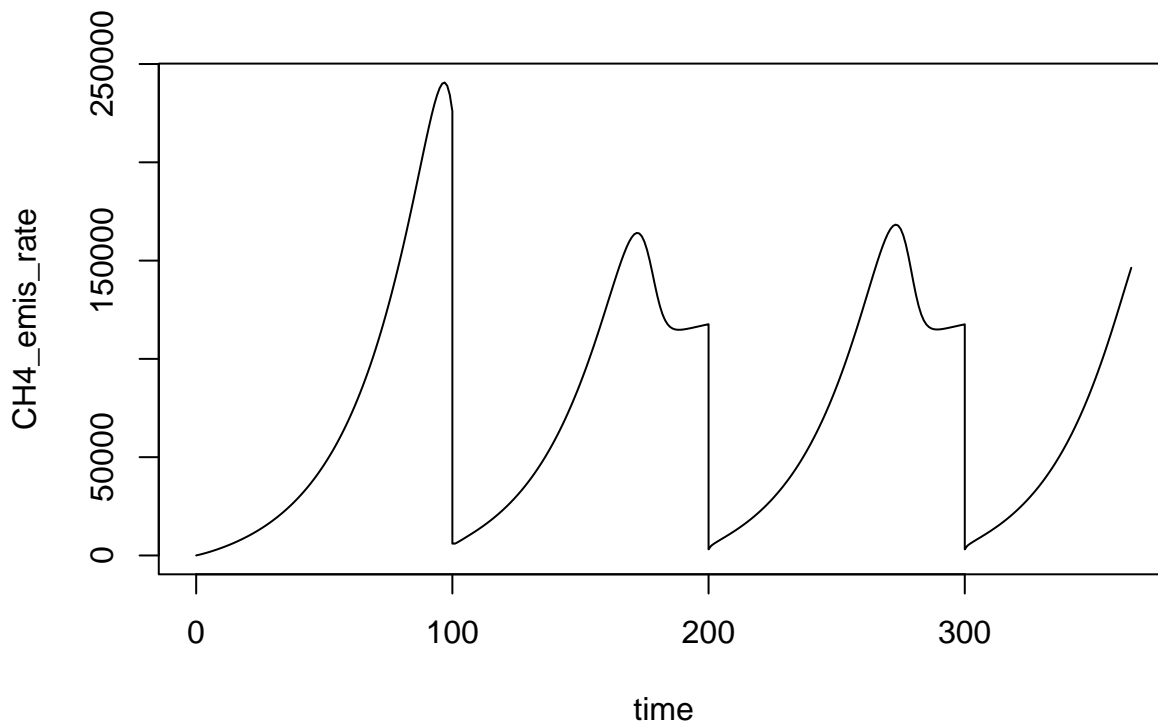
The effluent columns are cumulative. I did this for COD balance checking. We will have to discuss what is needed.

Here are some results.

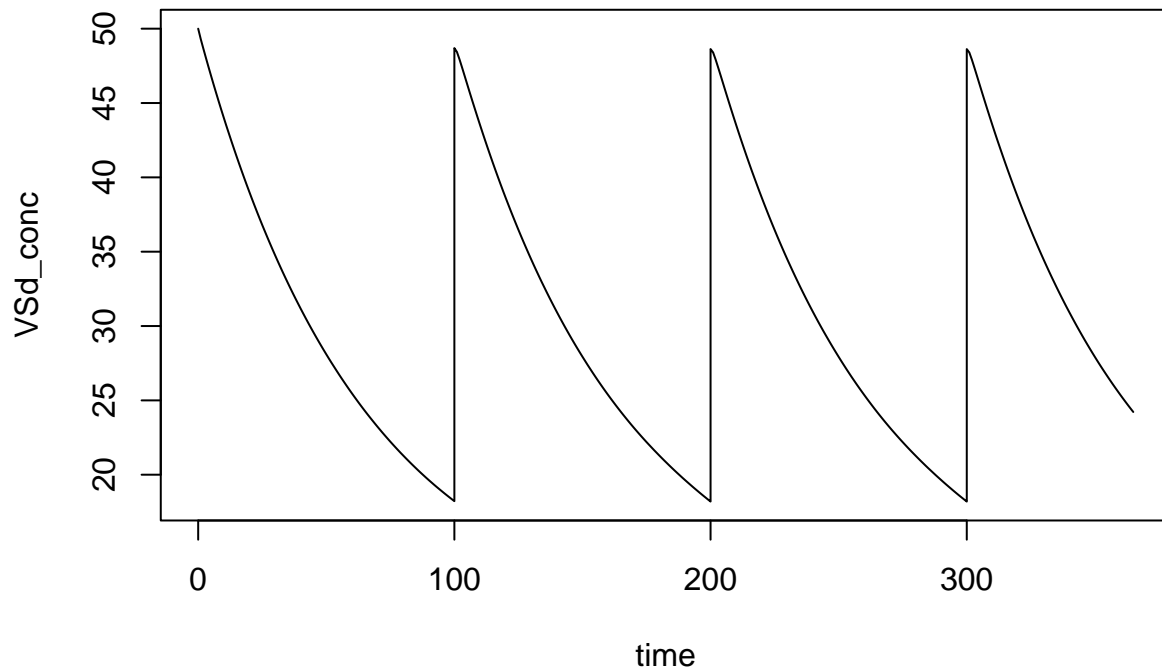
```
plot(slurry_mass ~ time, data = out1, type = 'l')
```



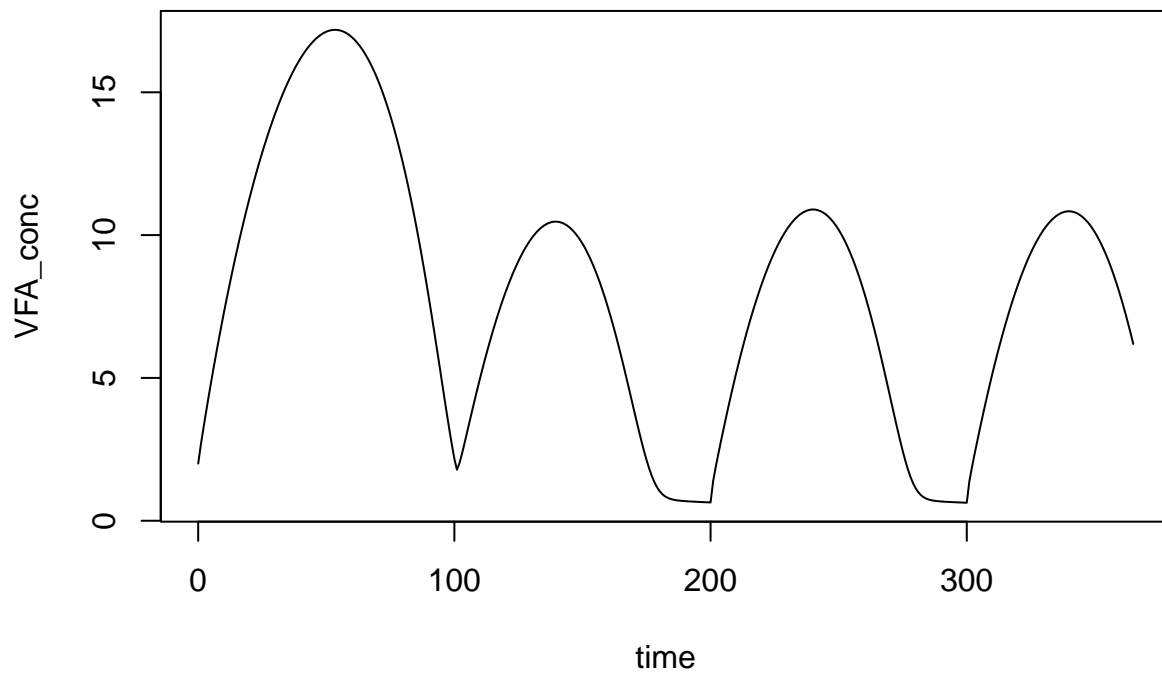
```
plot(CH4_emis_rate ~ time, data = out1, type = 'l')
```



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

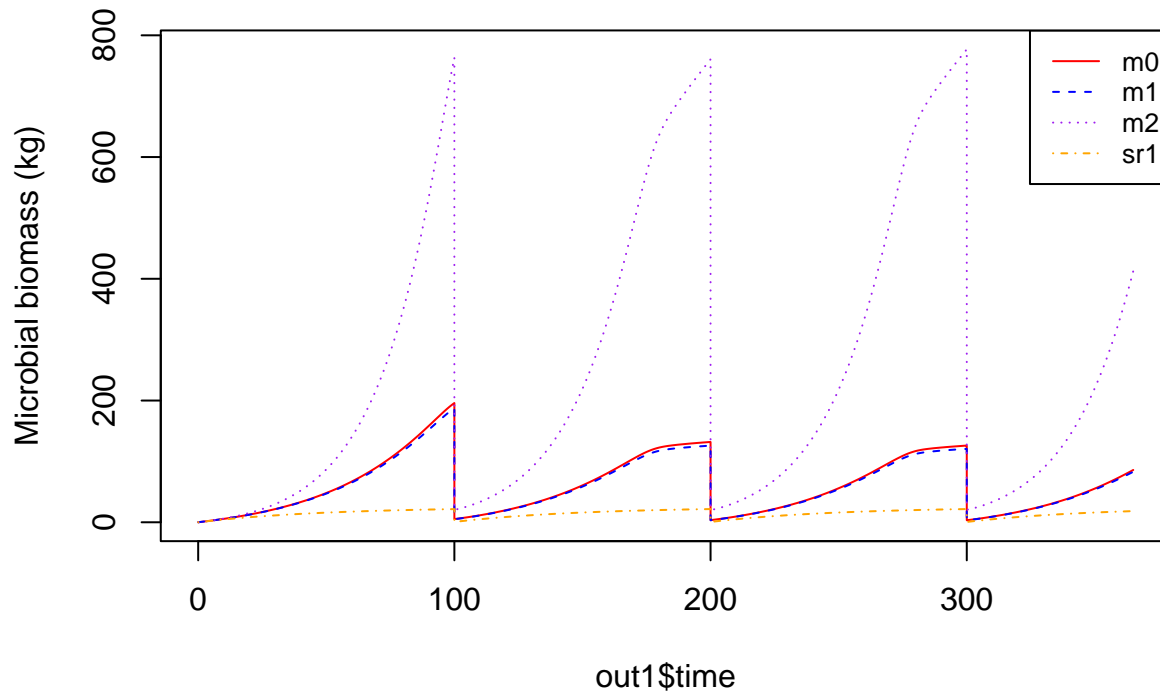


```
plot(VFA_conc ~ time, data = out1, type = 'l')
```



And methanogens.

```
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

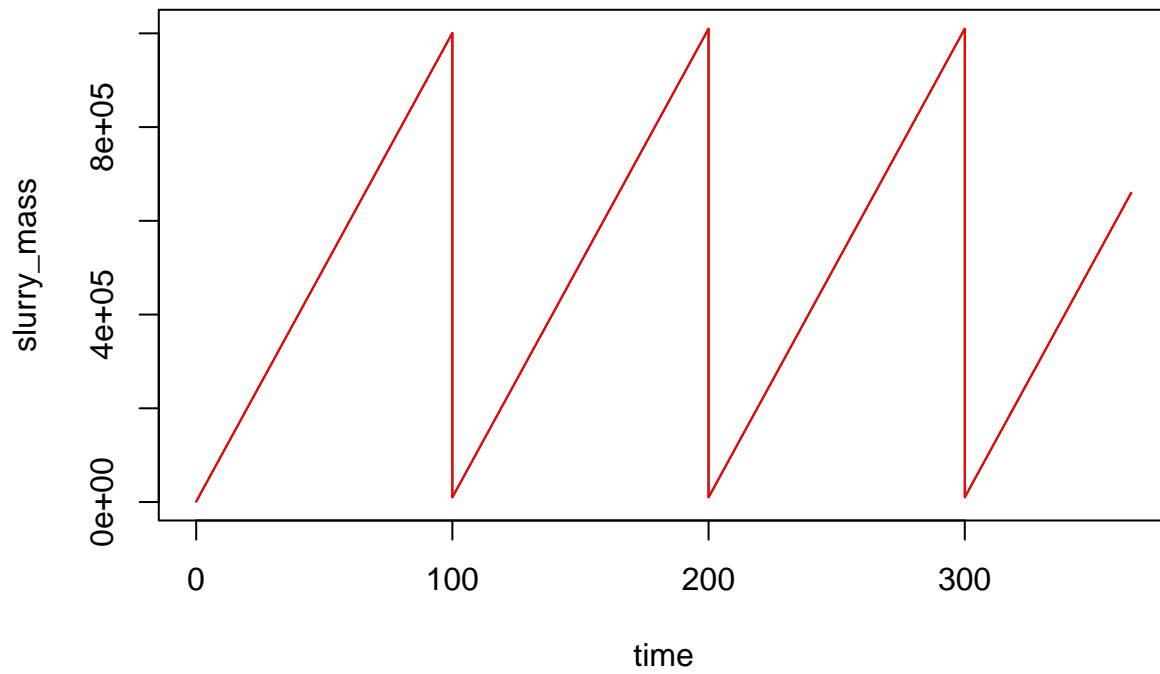
Particulate substrates are defined in `sub_pars` now and there are no particular substrates hard-wired in the code. VFA is the only intermediate, and it is hard-wired. Here we will include two more substrates. The mix of names might not make much sense in this example.

```
sub_pars2 <- list(subs = c('VSd', 'protein', 'cellulose'),
  T_opt_hyd = c(all = 60),
  T_min_hyd = c(all = 0),
  T_max_hyd = c(all = 90),
  hydrol_opt = c(VSd = 0.1, protein = 0.01, cellulose = 0.05),
  sub_fresh = c(VSd = 50, protein = 20, cellulose = 35),
  sub_init = c(VSd = 50, protein = 20, cellulose = 35))
```

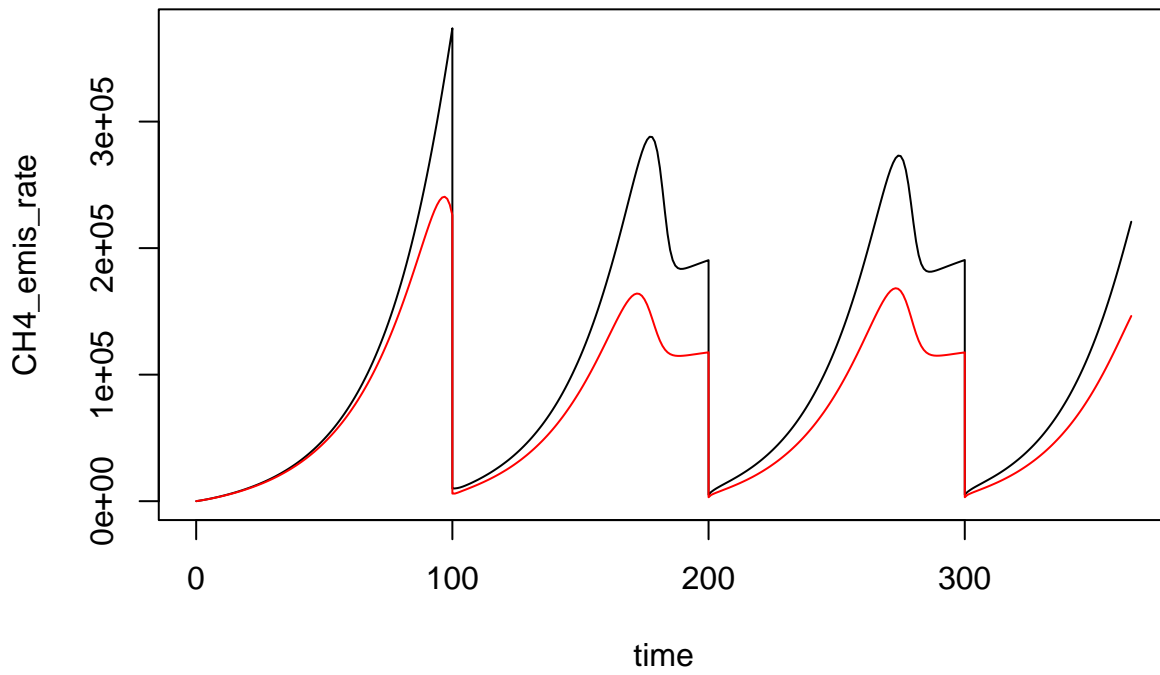
```
out2 <- abm(365,
  mng_pars = mng_pars,
  man_pars = man_pars,
  grp_pars = grp_pars,
  mic_pars = mic_pars,
  sub_pars = sub_pars2,
  chem_pars = chem_pars)
```

```
## Warning in checkCOD(dat = dat, grps = pars$grps, subs = pars$subs, COD_conv =
## pars$COD_conv, : COD balance is off by 1.1%
```

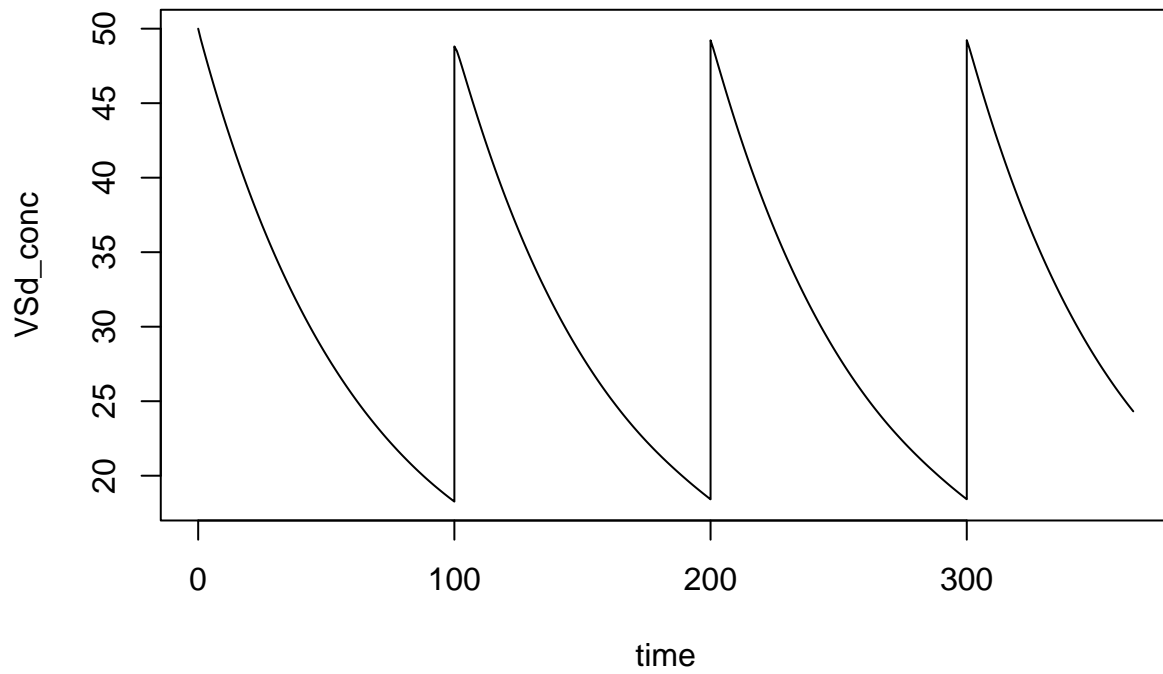
```
plot(slurry_mass ~ time, data = out2, type = 'l')
lines(slurry_mass ~ time, data = out1, col = 'red')
```



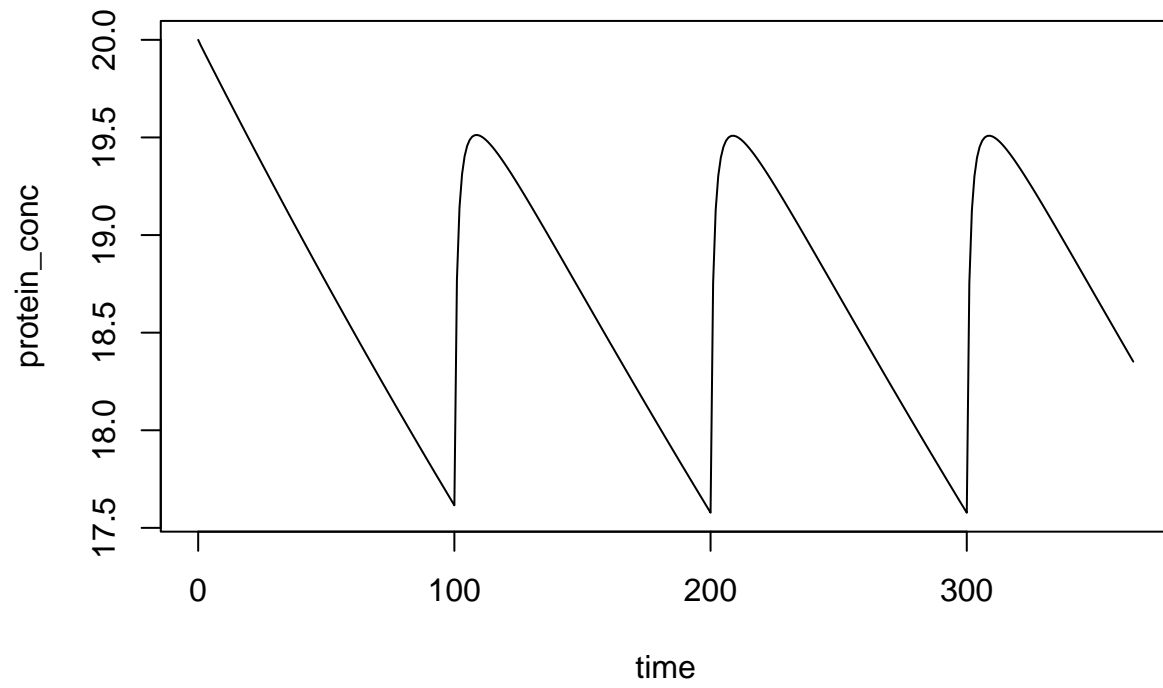
```
plot(CH4_emis_rate ~ time, data = out2, type = 'l')  
lines(CH4_emis_rate ~ time, data = out1, col = 'red')
```



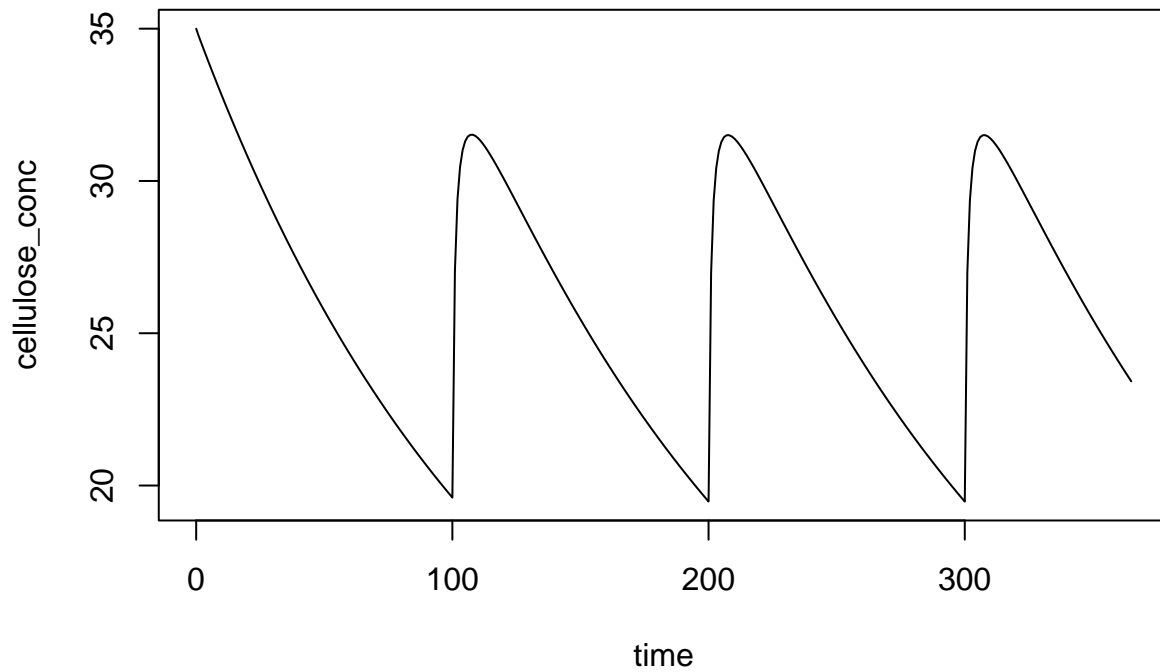
```
plot(VSd_conc ~ time, data = out2, type = 'l')
```

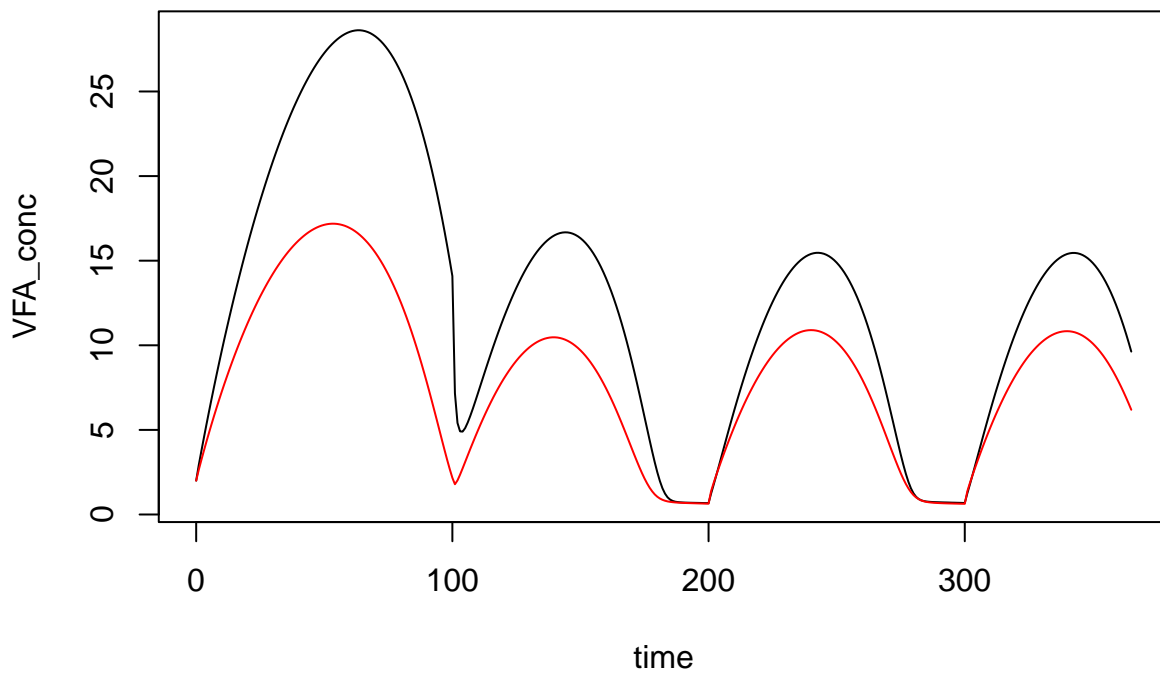
```
plot(protein_conc ~ time, data = out2, type = 'l')
```



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



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



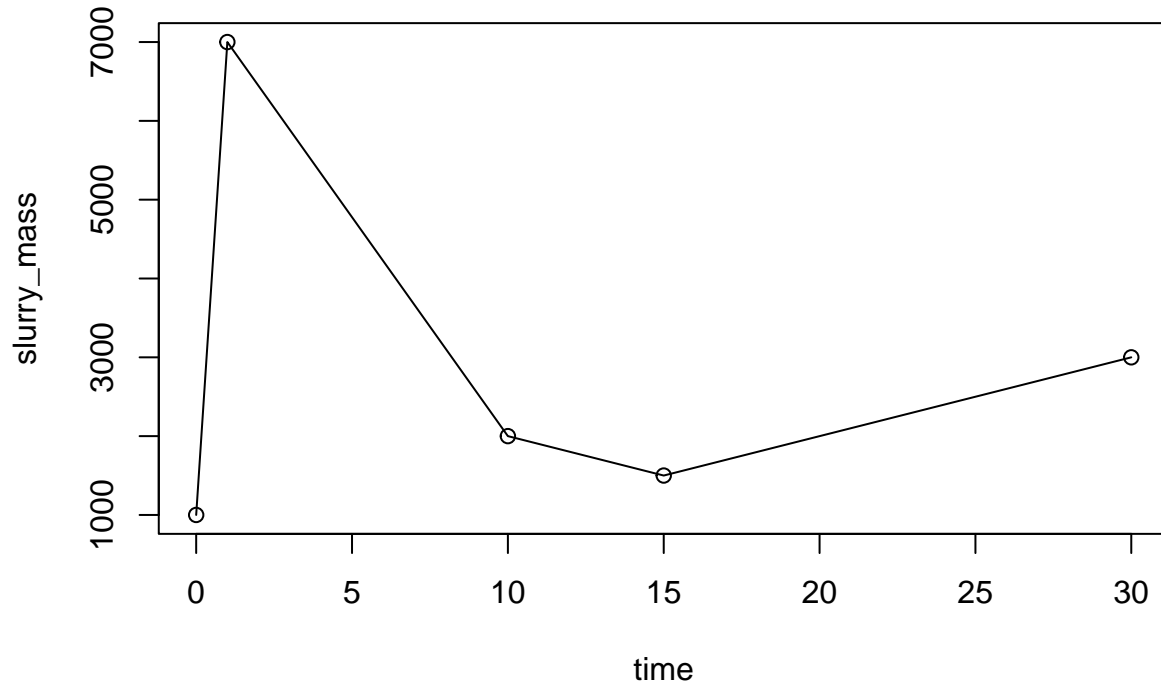
This flexibility comes from an approach similar to what we used for microbial groups.

3. Time-variable inputs part 1

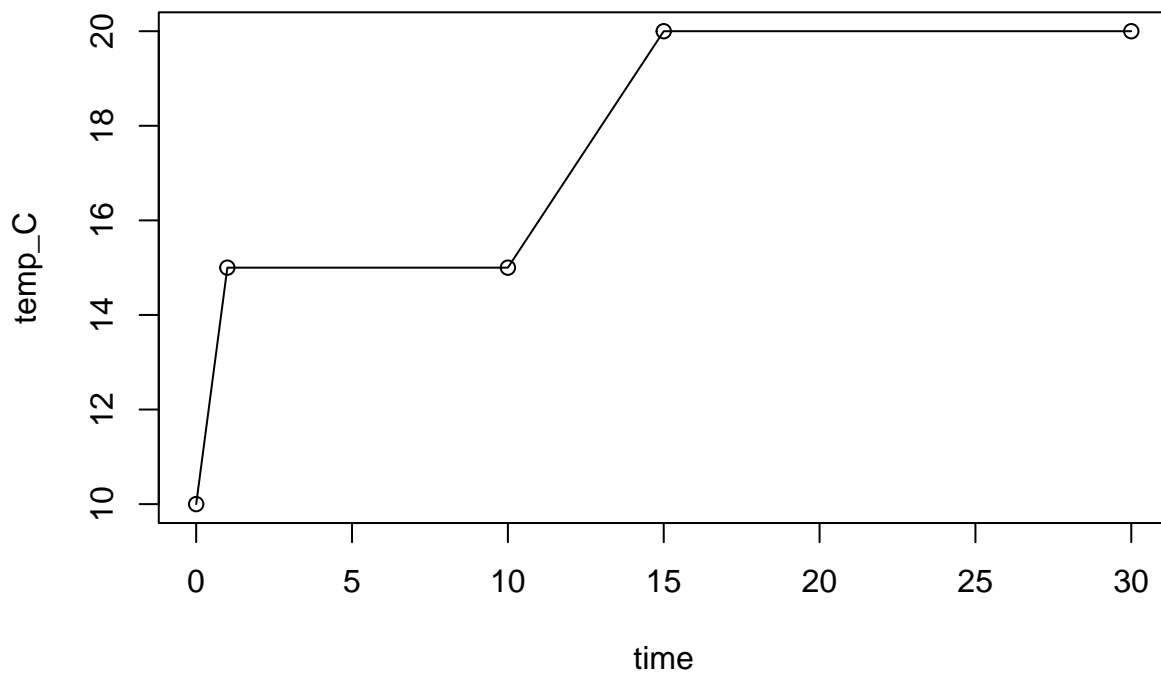
The `abm()` function can handle variability over time in any inputs now. Here slurry mass and temperature will vary.

```
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')
```



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

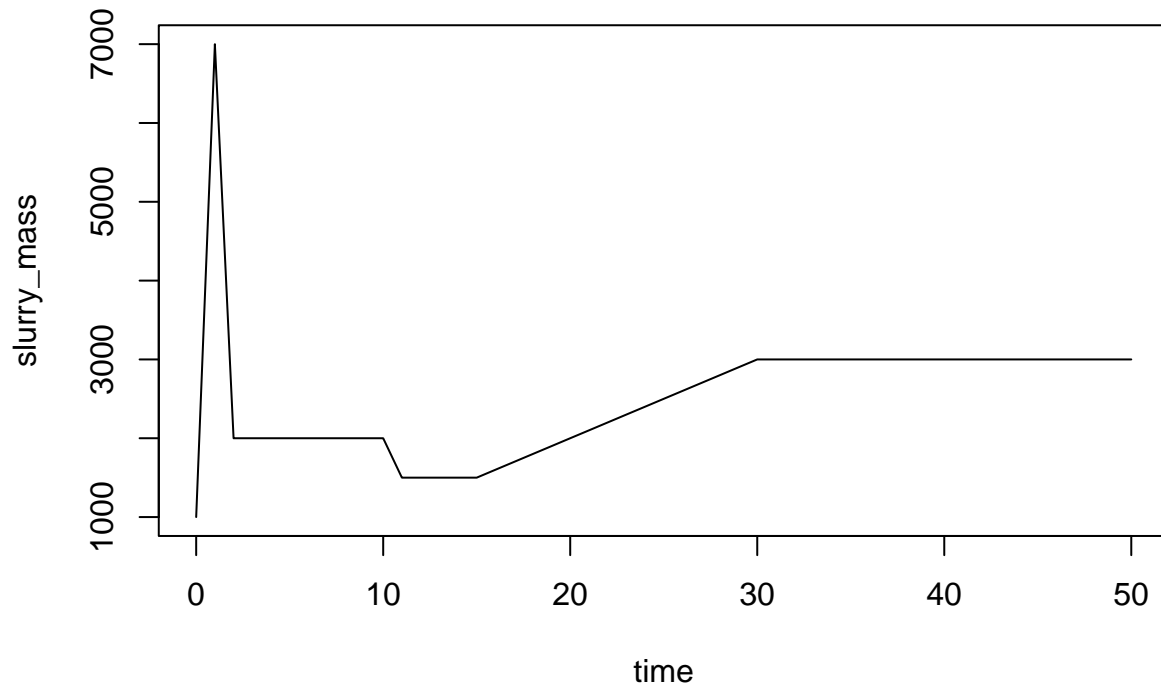


This data frame goes in the `var_pars` argument, which must be a list. The data frame must have a `slurry_mass` column if it is used—it is not possible to use an `abm_regular()`-like approach with variable temperature etc.

```
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 = out3, 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'))
```

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

```
## i Loading ABM
```

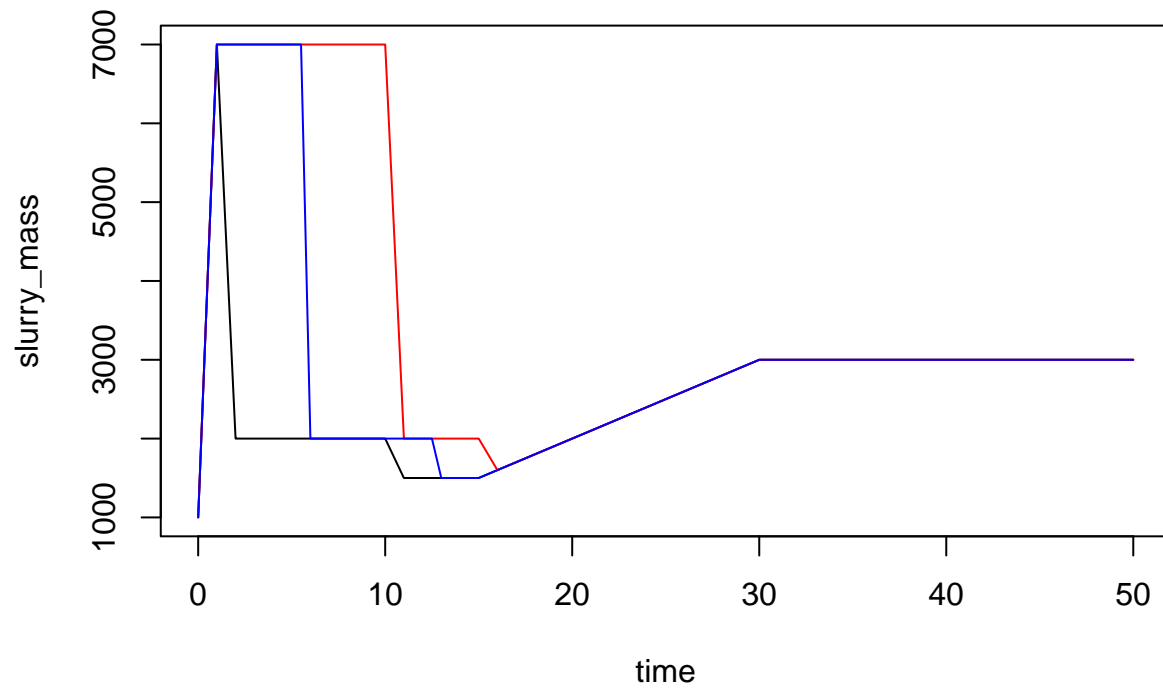
```
out3c <- abm(50,
  mng_pars = mng_pars,
  man_pars = man_pars,
  grp_pars = grp_pars,
```

```

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')

```

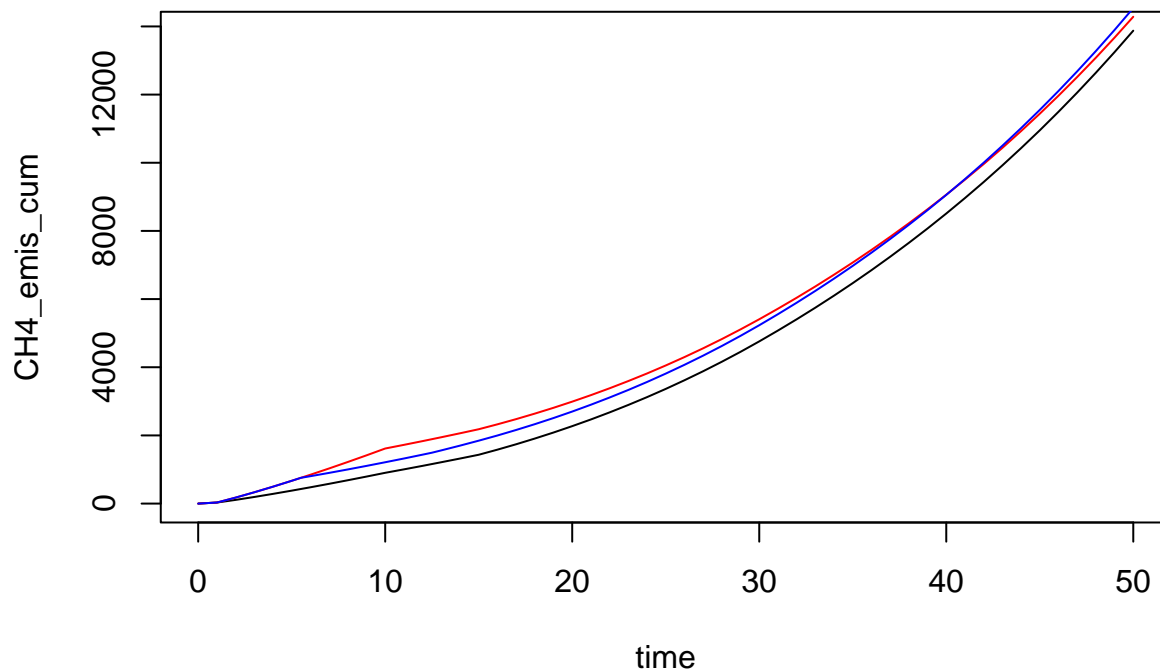


Here there is an effect of slurry mass and temperature.

```

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.

```
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
```

```
##   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         50
```

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

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

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

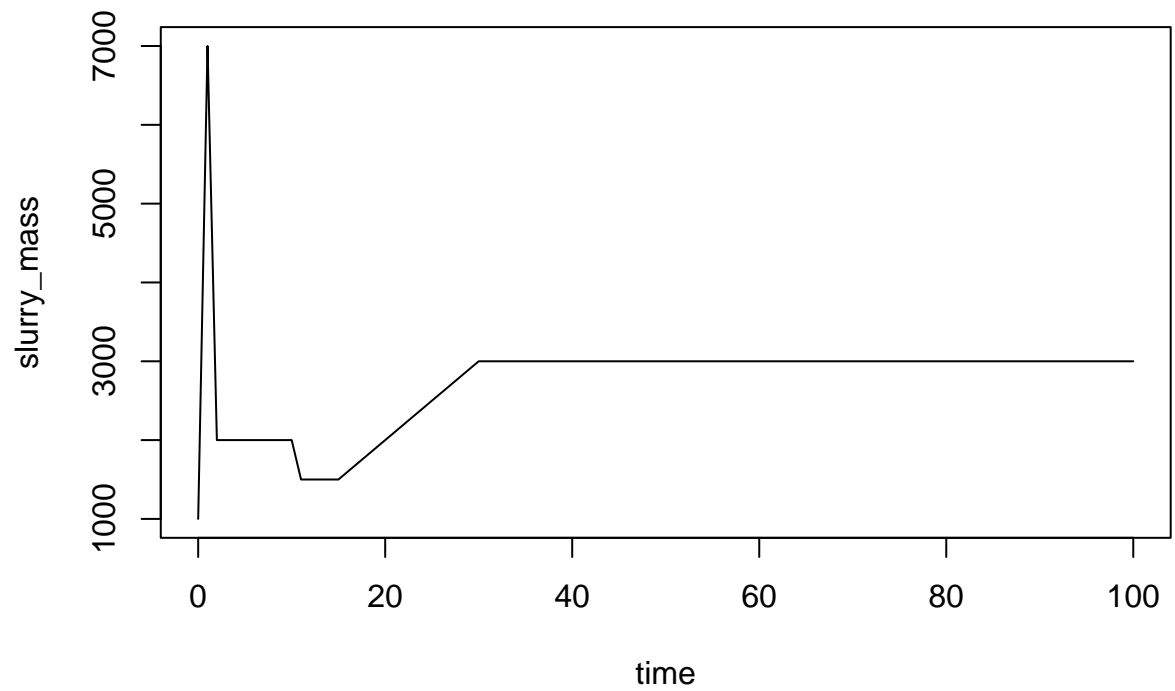
```
## [[1]]
## VSd
## 0
```

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

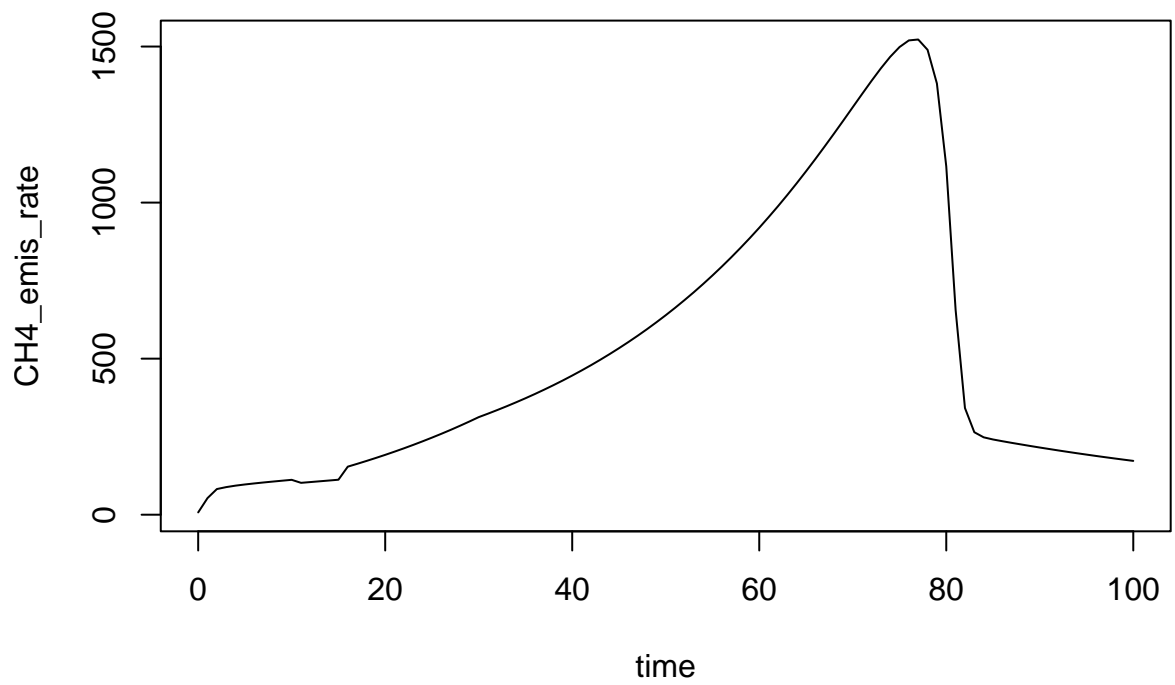
```
## i Loading ABM
```

```
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)
```

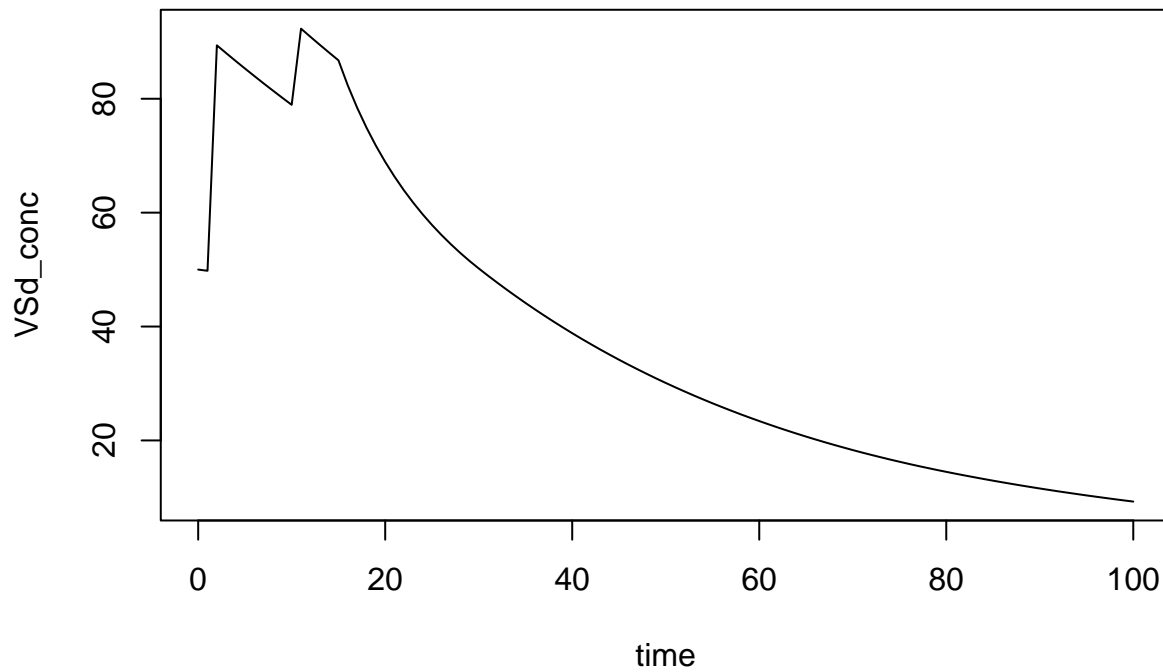
```
plot(slurry_mass ~ time, data = out4a, type = 'l')
```



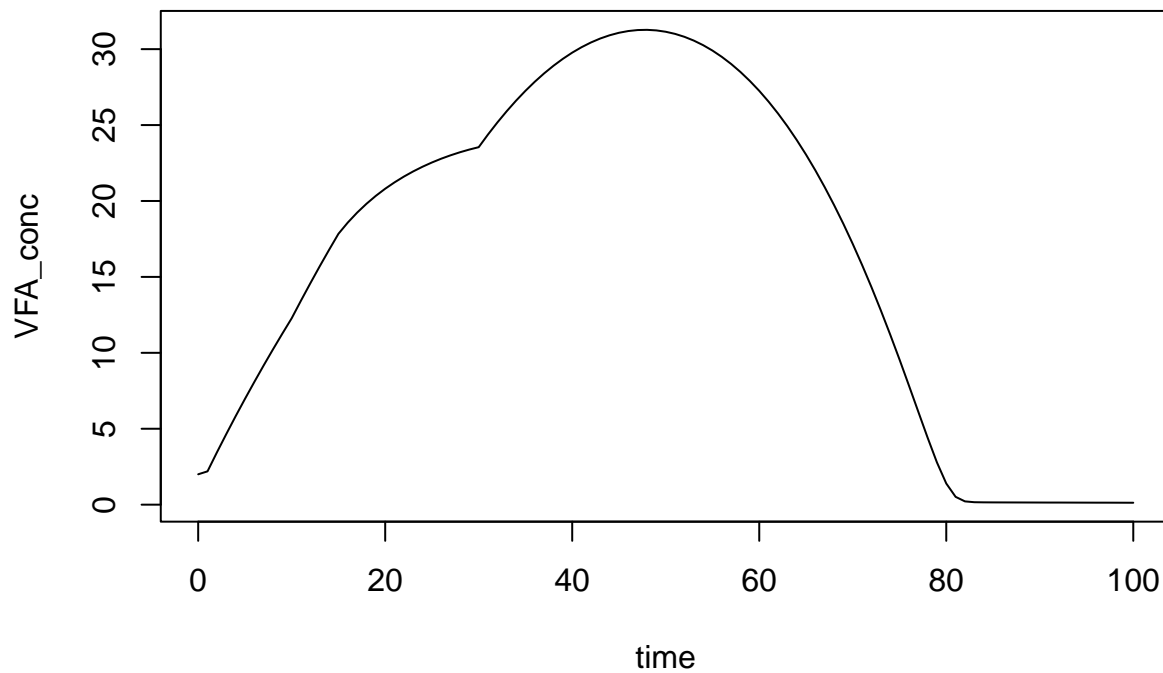
```
plot(CH4_emis_rate ~ time, data = out4a, type = 'l')
```



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

```
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))
```

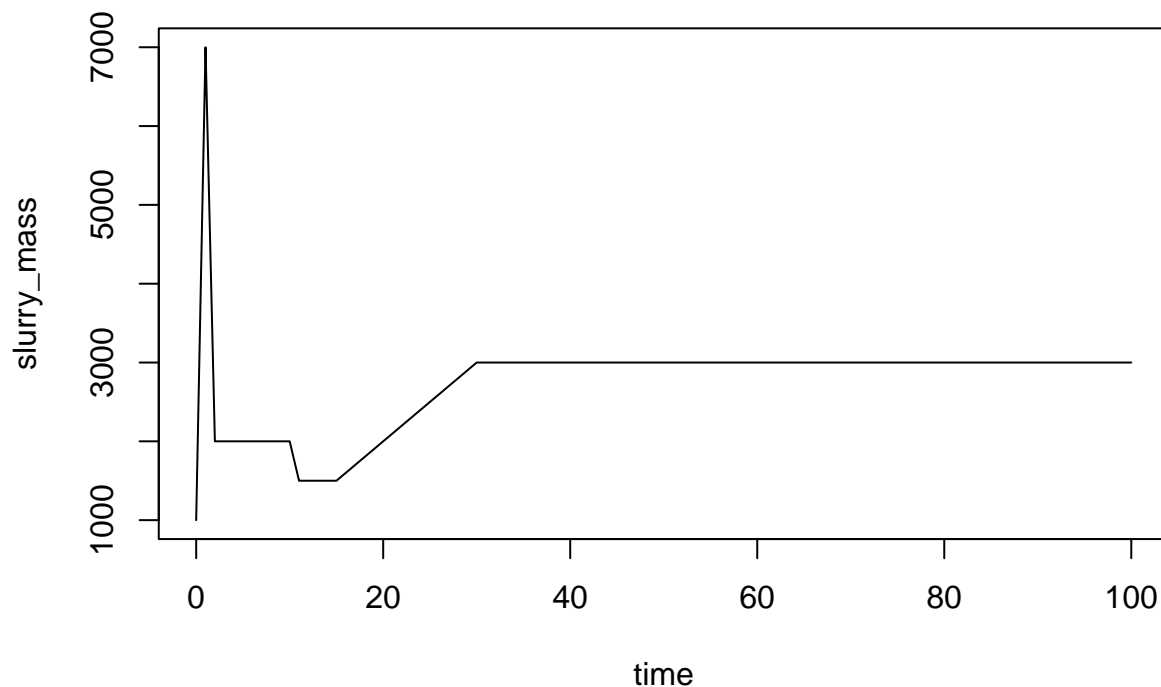
Some microbial parameters for a shift in temperature optima, “adaptation” for example.

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

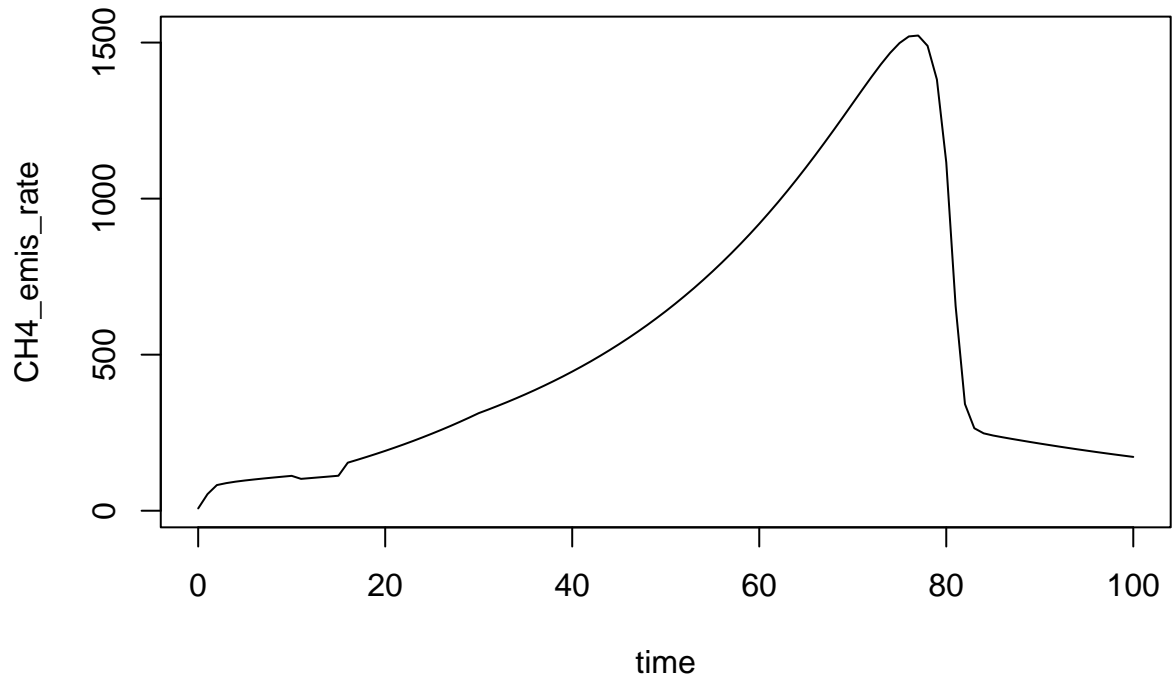
```
##    time slurry_mass  pH sub_fresh      T_opt
## 1     0      1000 7.0      50 20, 20, 30, 46
## 2     1      7000 6.9      50 22, 22, 32, 48
## 3    10      2000 6.8     100 24, 24, 34, 50
## 4    15      5000 6.7      50 26, 26, 36, 52
## 5    30      3000 6.6      50 28, 28, 38, 54
## 6    50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

```
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)
```

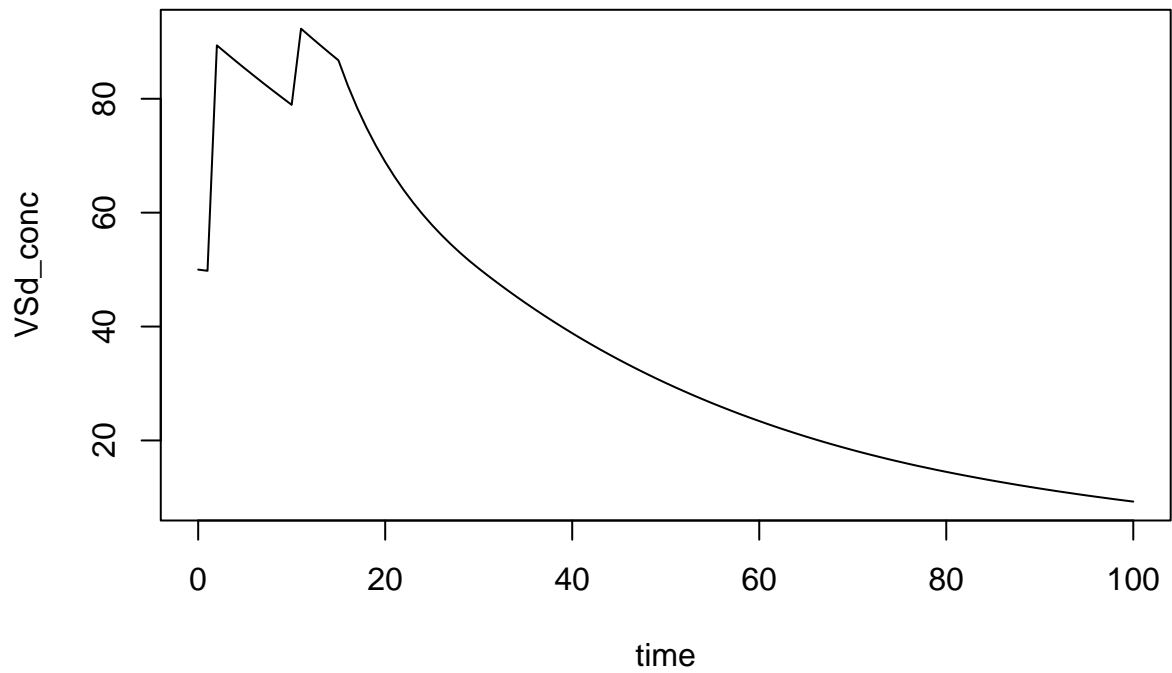
```
plot(slurry_mass ~ time, data = out4b, type = 'l')
```



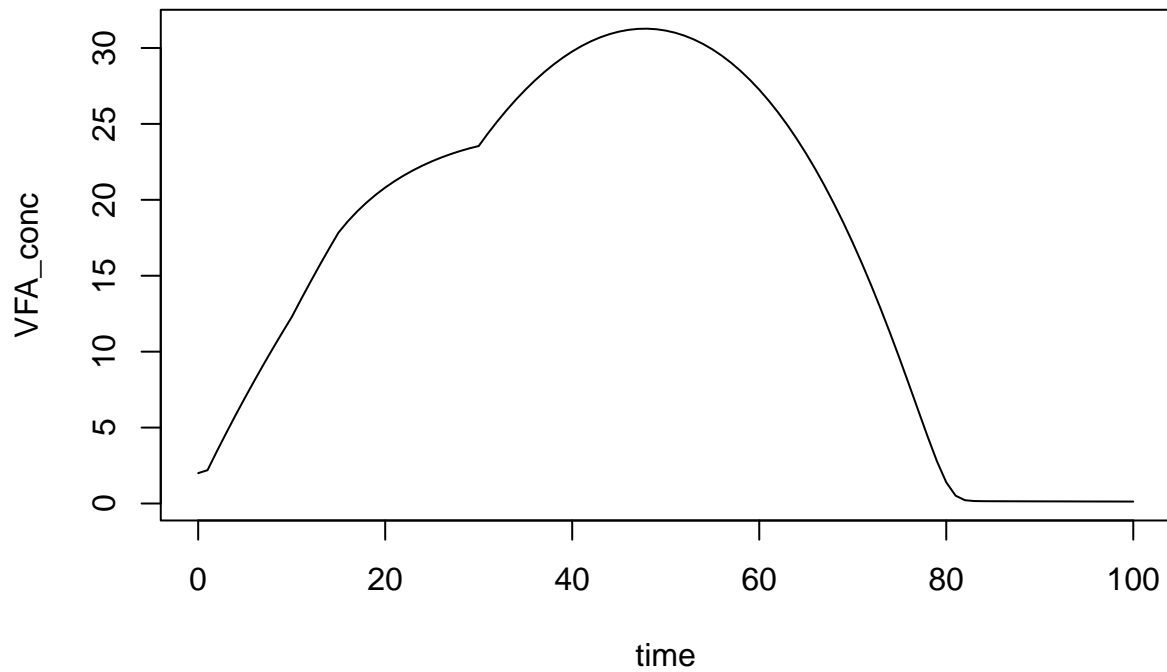
```
plot(CH4_emis_rate ~ time, data = out4b, type = 'l')
```



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



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



5. 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.