An introduction to the ATM99 model in R

```
options(width = 85)
```

Overview

The ATM99 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 **t**ransformation **m**odel, and 99 represents the unlimited number of microbial groups that can be included. 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. Although it was storage of organic waste (animal manure) in unheated tanks that drove the initial development of the model, with its flexibility it is well-suited to simulate biogas production from organic waste in anaerobic digesters, particularly in the presence of temperature variation. The purpose of this document is to demonstrate the use of the ATM99 R package, which is a flexible implementation of the model. The focus here is on the *use* of the `atm()` function; for a detailed description of the model itself, see Dalby et al. (2020a, 2020b).

Installation

The ATM99 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/ATM99')
```

And to use the ATM99 model, the package must be loaded.

```
library(ATM99)
```

REMOVE LATER

```
ff <- list.files('../R', full.names = TRUE)
for (i in ff) source(i)
ls()
```

```
##  [1] "atm"          "atm_regular"  "atm_variable" "ff"           "H2SO4_titrat"
##  [6] "i"            "logistic"     "logit"        "nn"           "out4"
## [11] "out5"         "pH_fun"       "pred1"        "pred1s"       "pred2"
## [16] "pred2b"       "pred3"        "pred3b"       "rates"        "SO4_fun"
## [21] "temp_C_fun"   "temp_dat"
```

A simple example

By default, the `atm()` function simulates degradation of animal manure from a 33 m^3 storage tank with a 30 day emptying interval. Fresh slurry is added continuously at a rate of 1000 kg d^{-1}. 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).

```
out1 <- atm()
```

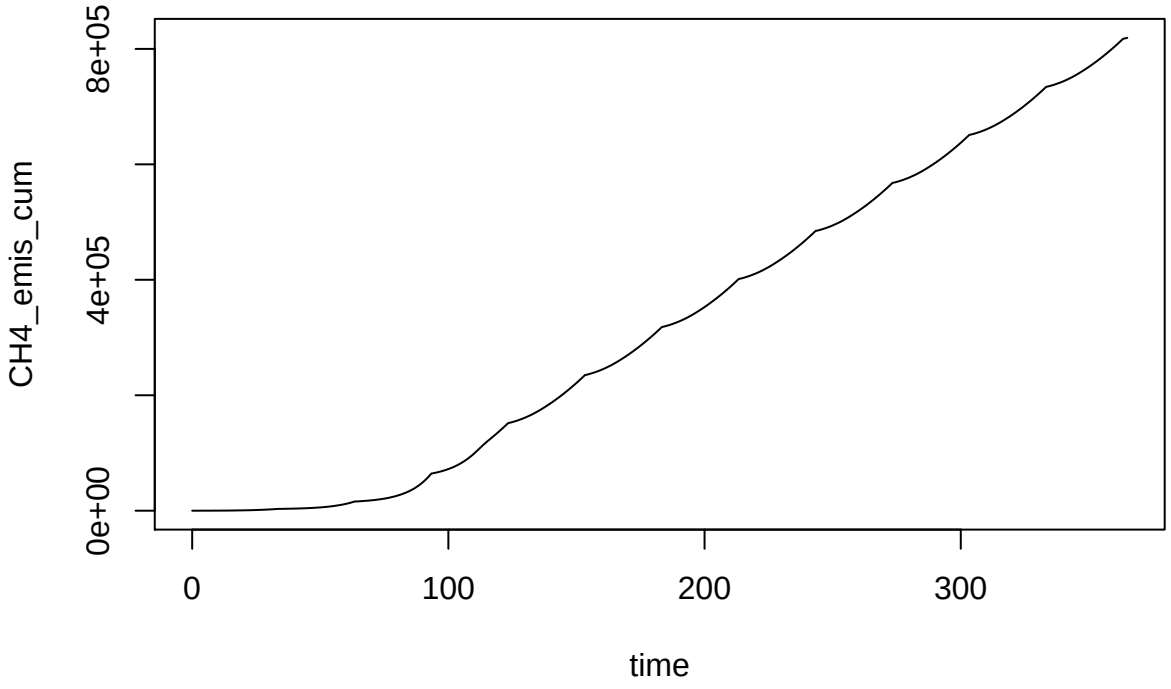
Output is, by default, a data frame with predicted variables over time. Typically the primary variable of interest is CH_4 emission, which is returned as a total (g) and rate, overall or normalized to COD or VS mass:

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

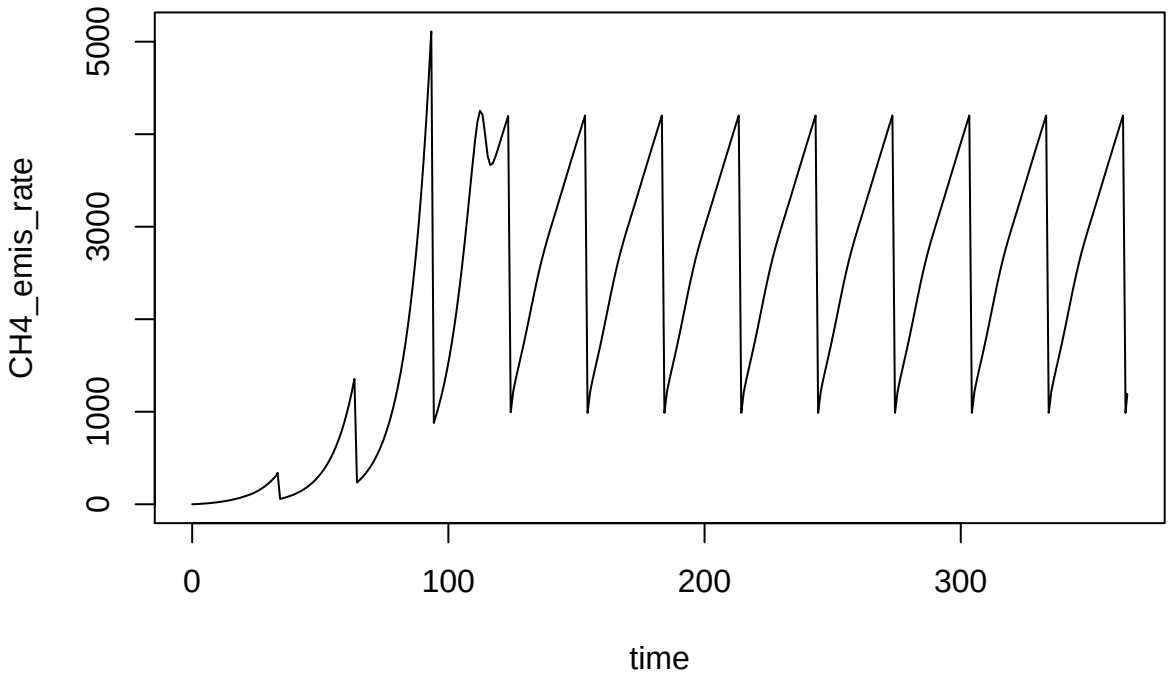
```
##  [1] "CH4_emis_cum"         "CH4_emis_rate"        "CH4_emis_rate_slurry"
##  [4] "CH4_flux"             "CH4_emis_rate_COD"    "CH4_emis_rate_dCOD"
##  [7] "CH4_emis_rate_VS"     "CH4_emis_cum_COD"     "CH4_emis_cum_dCOD"
## [10] "CH4_emis_cum_VS"
```

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

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

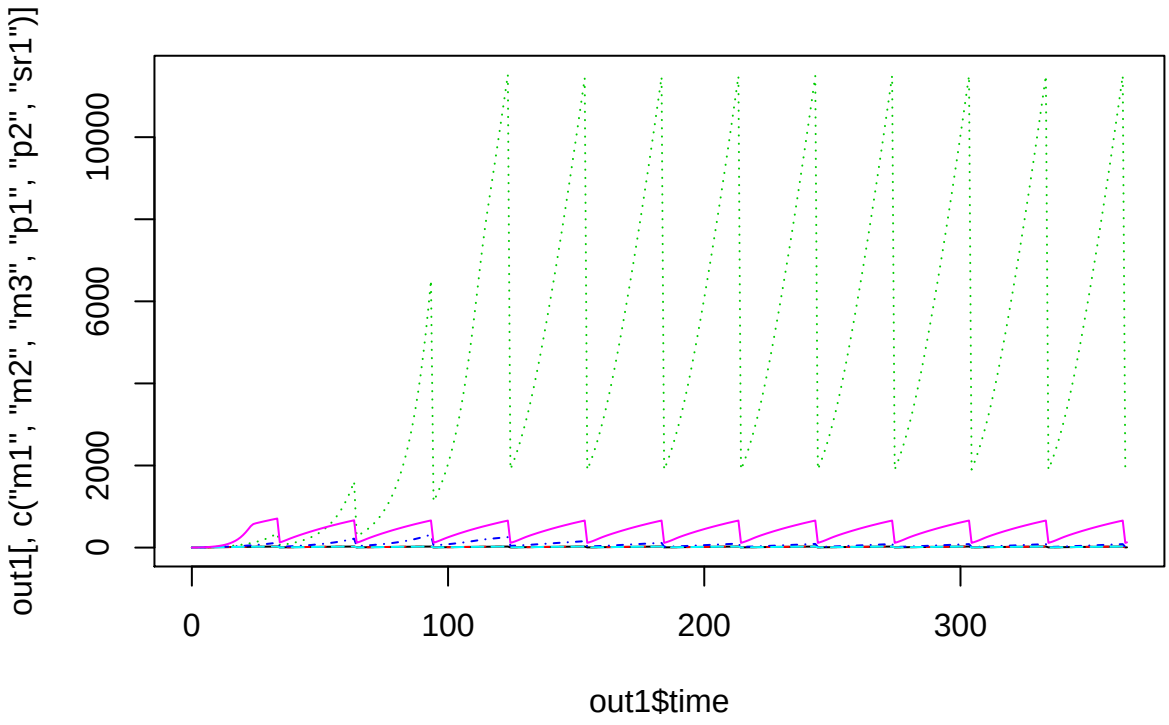



```
plot(CH4_emis_rate ~ time, data = out1, type = 'l')
```



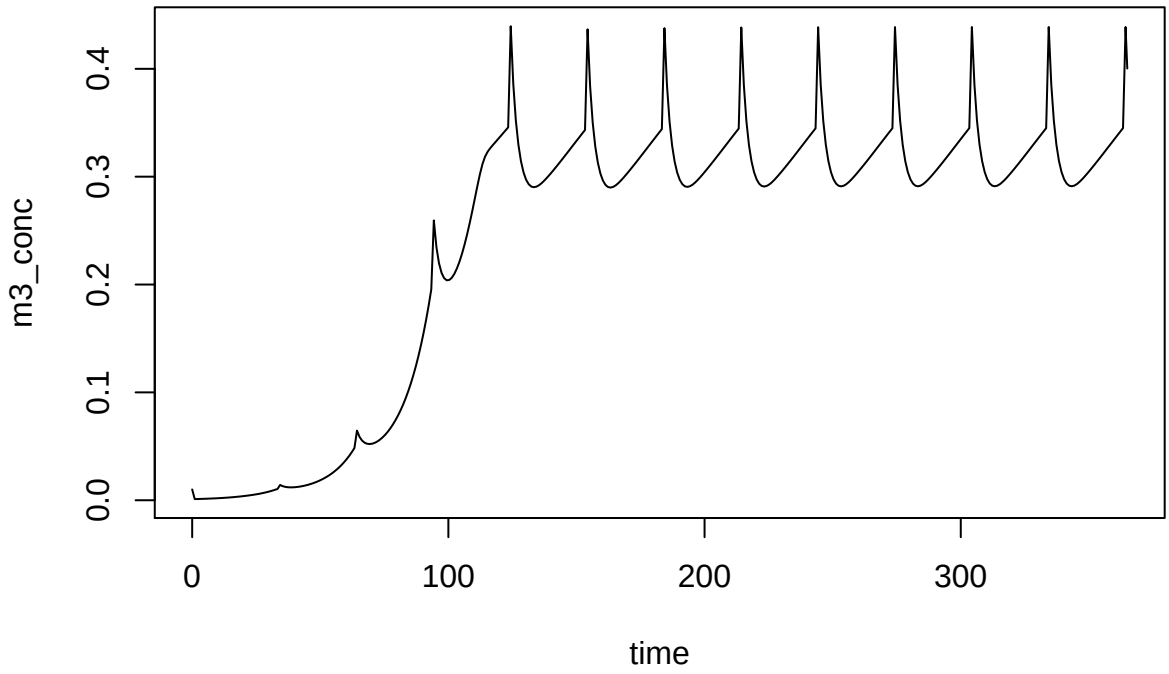
Microbial biomass (g) is given in columns with the names set in the `grp_pars` argument.

```
matplot(out1$time, out1[, c('m1', 'm2', 'm3', 'p1', 'p2', 'sr1')], type = 'l')
```



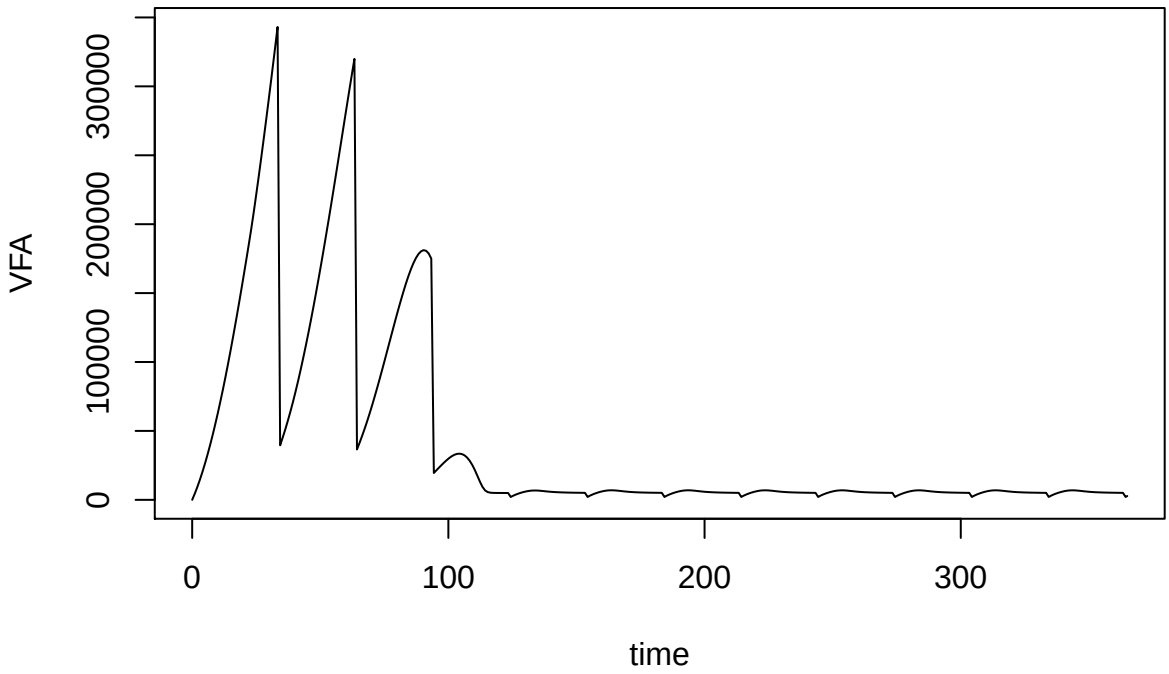
Because of a default temperature of 23 (NTS: why so high???) methanogen `m3` dominates. Biomass concentrations (g/kg) may be more informative.

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

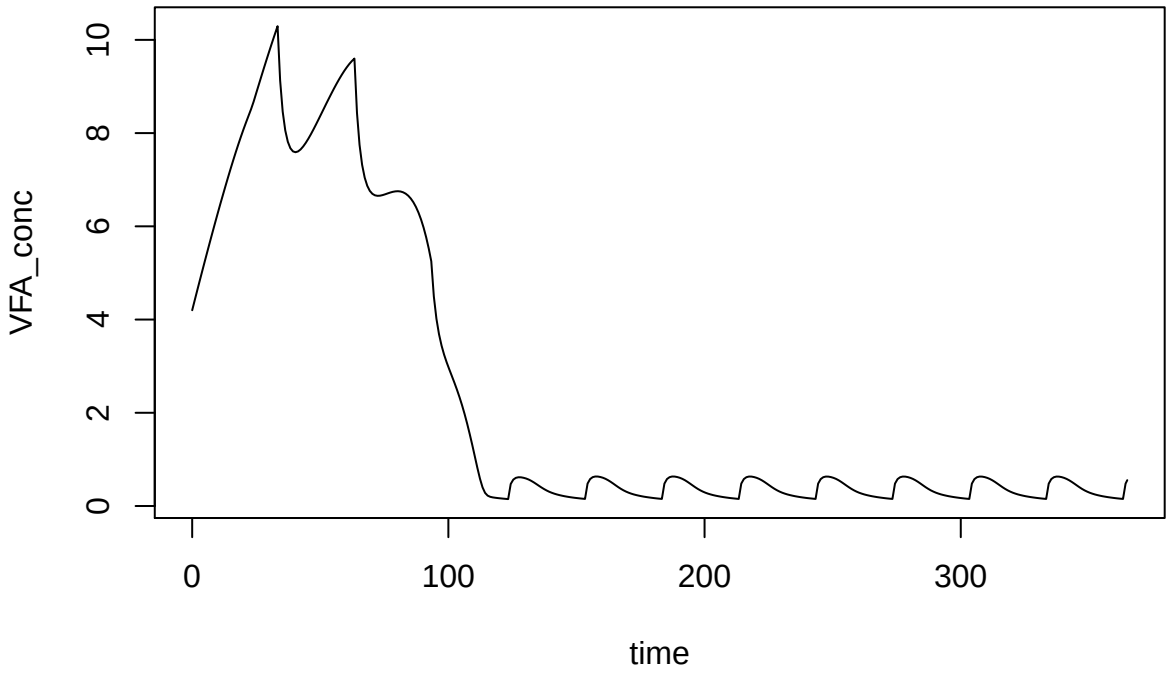


Dynamics in production of CH_4 are often related to VFA accumulation, and VFA mass (g) and concentration (g/kg) can be extracted. For more information on the many output variables returned by `atm()`, see the section on that topic below.

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



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



Setting parameter values

Although the ATM99 model is relatively simple, explicitly simulating the activity of multiple microbial groups means there are a lot of parameters. The complete list can be seen in the help file.

```
?atm
```

Parameters are grouped to make changes easier (or to prevent mistakes) and to limit the number of parameter names that are needed. But there are also some shortcuts built into the `atm()` funtion to make small tweaks simple.

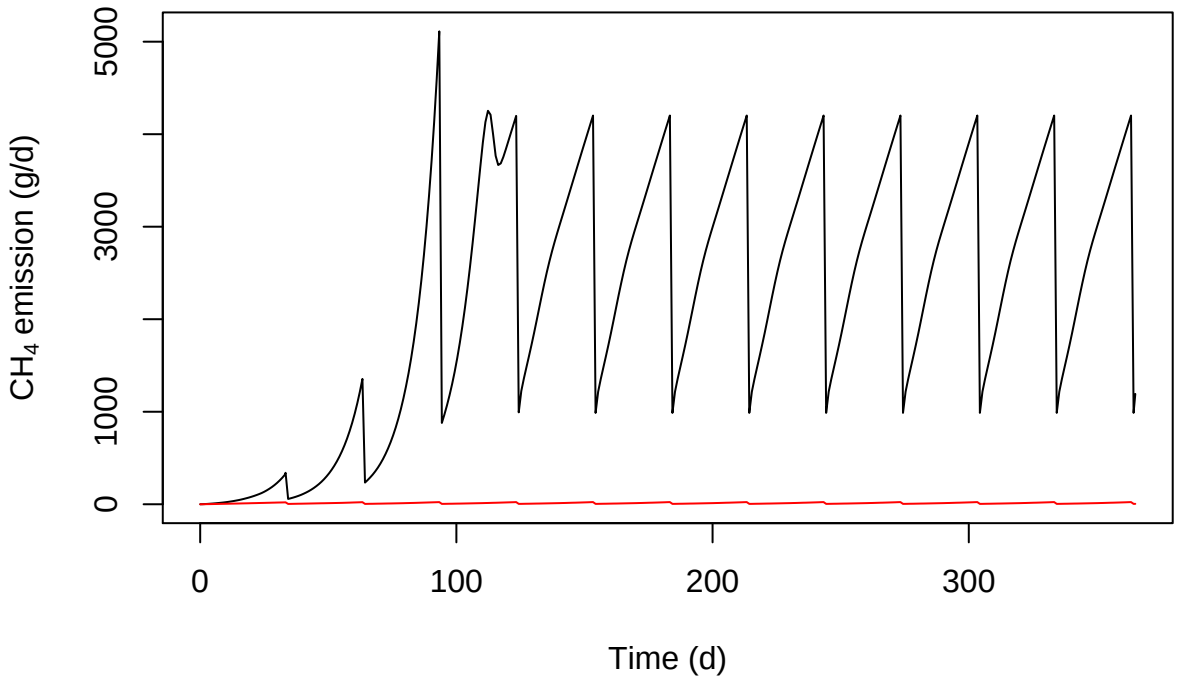
As an example, the composition of the fresh slurry (influent, or feed) is set with the `man_pars` argument, which is a list of solute concentrations and pH. By default:

```
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:

```
out2 <- atm(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))
```

```
plot(CH4_emis_rate ~ time, data = out1, type = 'l', xlab = 'Time (d)', ylab = expression('CH'[4]~'emiss
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.

```
out2b <- atm(365, 1, add_pars = list(pH = 6))
all.equal(out2, out2b)
```

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

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

Many arguments for the `atm()` 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` use the following call.

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

This shortcut is referred to as the "par.element" approach, 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 selected with the `par_key` argument.)

Of course, specifying all elements is always an option,

```
out3b <- atm(365, 1, add_pars = list(pH = 6, conc_fresh = list(S2 = 0.0, SO4 = 0.2, TAN = 1.0, VFA = 10
```

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.

Output options

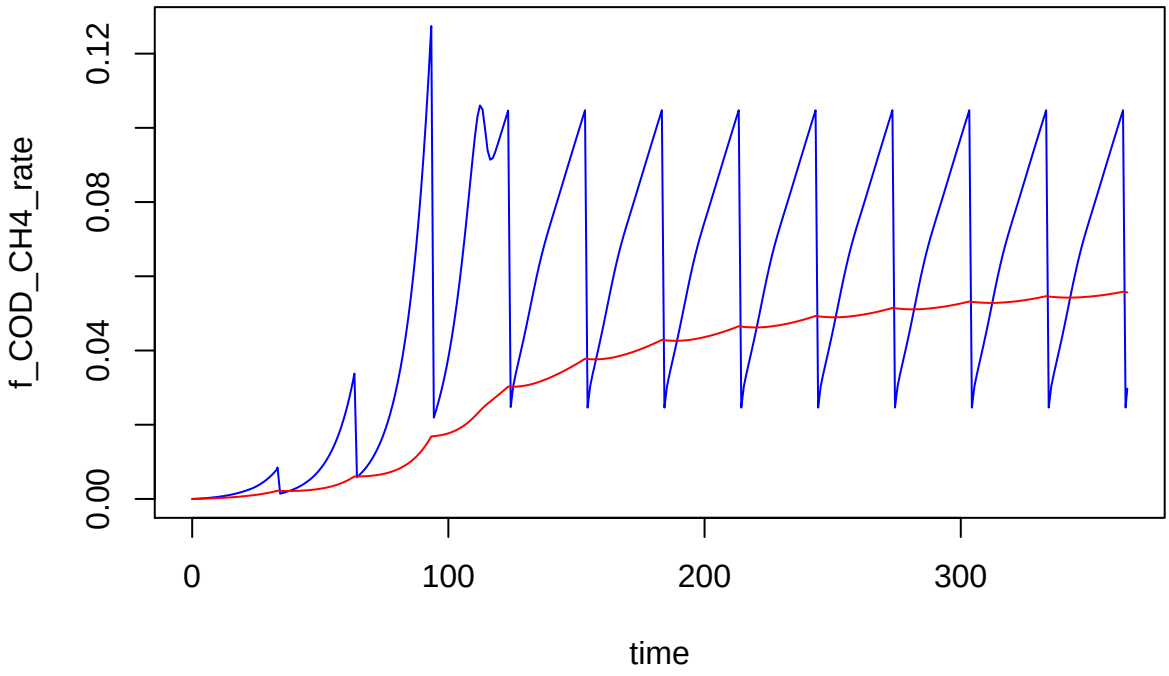
By default, the `atm()` function returns a data frame with cumulative CH_4 emission and other state variables, normalized in a variety of ways. In total there are more than 300 columns—the first 20 are shown below.

```
out1 <- atm(365, 1)
out1[1:3, 1:20]
```

```
##   time         m1         m2         m3         p1         p2        sr1 slurry_mass
## 1    0 0.00000001 0.00000001 0.00000001 0.00000001 0.00000001 0.00000001       1e-06
## 2    1 0.99089164 0.99144403 1.05920382 1.03470420 0.99006634 1.11460463       1e+03
## 3    2 1.96378854 1.96597992 2.24656139 2.14223790 1.96052805 2.49416779       2e+03
##             Sp          VFA     sulfate   sulfide CH4_emis_cum CO2_emis_cum
## 1 6.500000e-05    0.0000042   0.0000002 0.0000000    0.0000000       0.0000
## 2 6.473805e+04 4416.2312952 199.0328394 0.9547404    0.7317084      57.3837
## 3 1.290361e+05 9261.9255150 395.8297136 4.0661864    3.0254742     124.1151
##   COD_conv_cum COD_conv_cum_meth COD_conv_cum_respir COD_conv_cum_sr       NH4
## 1      0.00000          0.000000             0.00000        0.000000 0.9950865
## 2     45.78025          2.918661            40.93305        1.928536 0.9950865
## 3    102.24983         12.068106            81.86610        8.315626 0.9950865
##          NH3
## 1 0.00491348
## 2 0.00491348
## 3 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. The fraction of loaded COD converted through methanogenesis, respiration, and sulfate reduction is also given. 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).

```
out1s <- atm(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    8.192569e+05
##   CH4_emis_rate    CH4_emis_COD   CH4_emis_dCOD     CH4_emis_VS    CO2_emis_cum
##    2.244539e+03    1.395192e-02    3.225598e-02    2.022018e-02    2.069105e+06
```

```
##   CO2_emis_rate    CO2_emis_COD   CO2_emis_dCOD     CO2_emis_VS   COD_conv_meth
##    5.668782e+03    3.523681e-02    8.146533e-02    5.106784e-02    3.267877e+06
## COD_conv_respir     COD_conv_sr       f_COD_CH4    f_COD_respir        f_COD_sr
##    1.494056e+04    1.438404e+05    5.565186e-02    2.544374e-04    2.449599e-03
```

And an arbitrary startup period can be excluded using the `startup` argument. For example, results are based on the last 165 days in the example below.

```
out1s <- atm(365, 1, value = 'sum', startup = 200)
out1s
```

```
##        COD_load       dCOD_load      ndCOD_load         VS_load    CH4_emis_cum
##    2.656000e+07    1.148820e+07    1.507180e+07    1.832640e+07    4.654488e+05
##   CH4_emis_rate    CH4_emis_COD   CH4_emis_dCOD     CH4_emis_VS    CO2_emis_cum
##    2.826581e+03    1.752443e-02    4.051540e-02    2.539772e-02    1.151153e+06
##   CO2_emis_rate    CO2_emis_COD   CO2_emis_dCOD     CO2_emis_VS   COD_conv_meth
##    6.990730e+03    4.334161e-02    1.002031e-01    6.281392e-02    1.856597e+06
## COD_conv_respir     COD_conv_sr       f_COD_CH4    f_COD_respir        f_COD_sr
##    6.740384e+03    6.471570e+04    6.990199e-02    2.537795e-04    2.436585e-03
```

Set the `value` argument to `'all'` for time series data and the summary.

Defining microbial groups

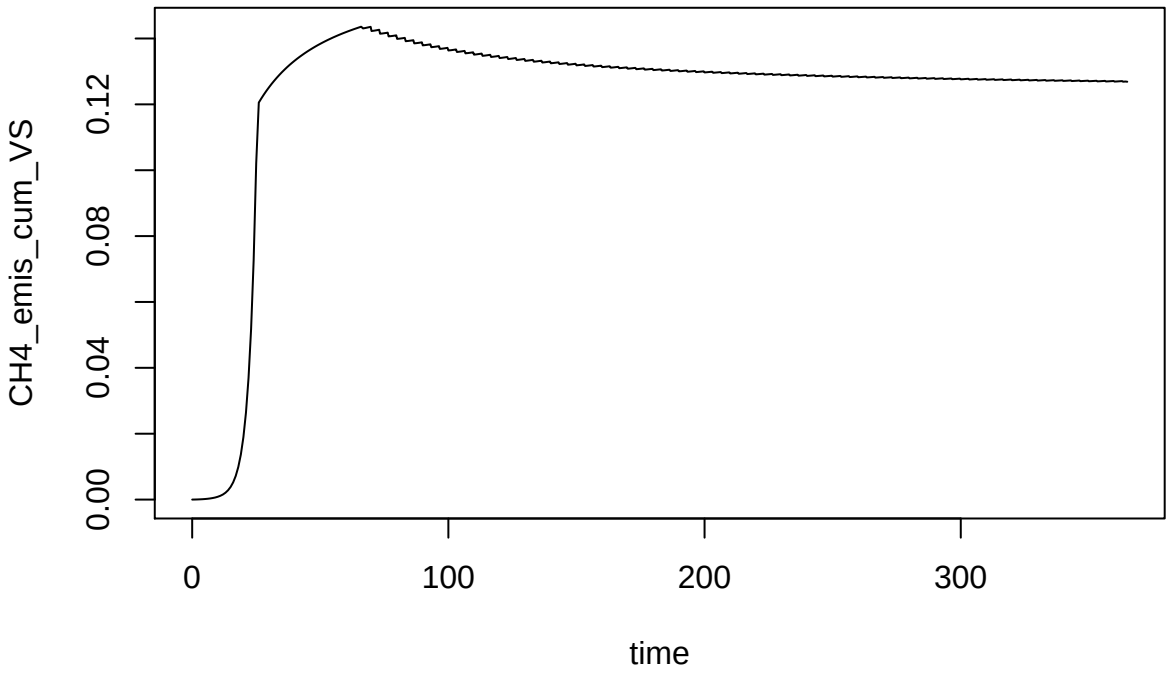
Simulating reactors

The ATM99 model inherently describes a reactor with continuous feeding and intermittent wasting. To approximate a continuous reactor (which typically has intermittent feeding and wasting in practice–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).

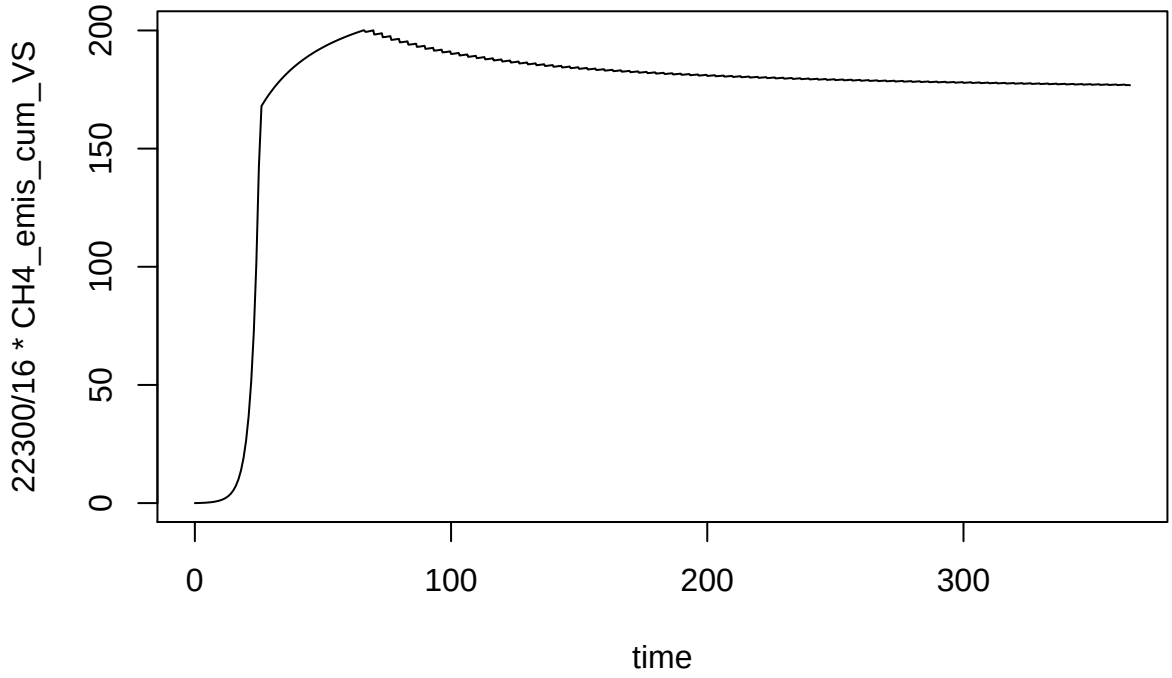
```
out4 <- atm(365, 1, add_pars = list(temp_C = 35, resid_frac = 0.95, alpha_opt = 0.2, slurry_prod_rate =
```

Due to the structure of the code, one drawback of this high `resid_frac` approach is a long evaluation time.

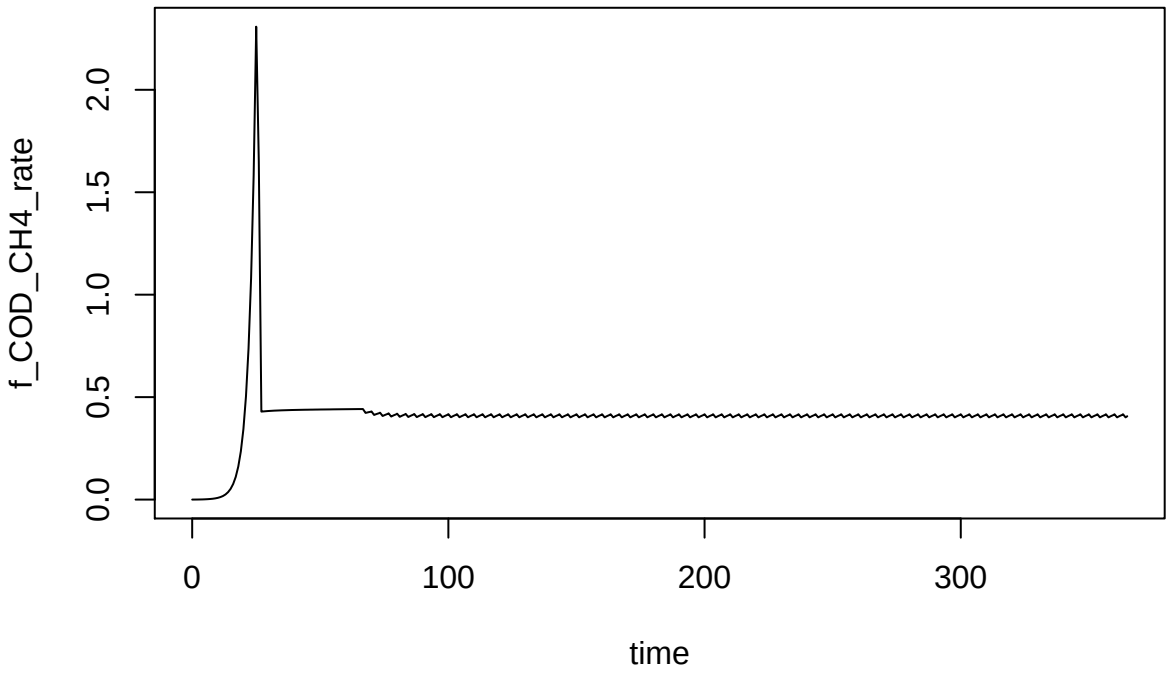
```
plot(CH4_emis_cum_VS ~ time, data = out4, type = 'l')
```



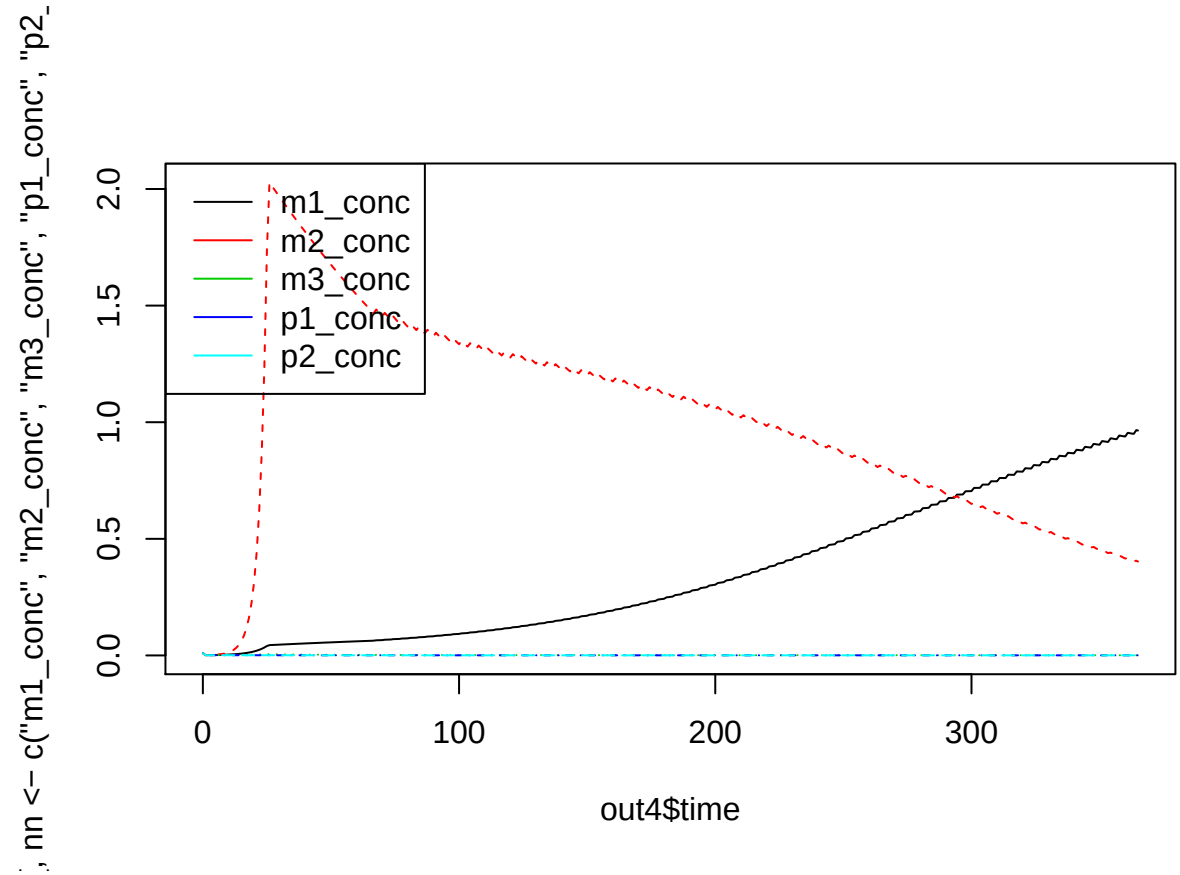

```
plot(22300 / 16 * CH4_emis_cum_VS ~ time, data = out4, type = 'l')
```



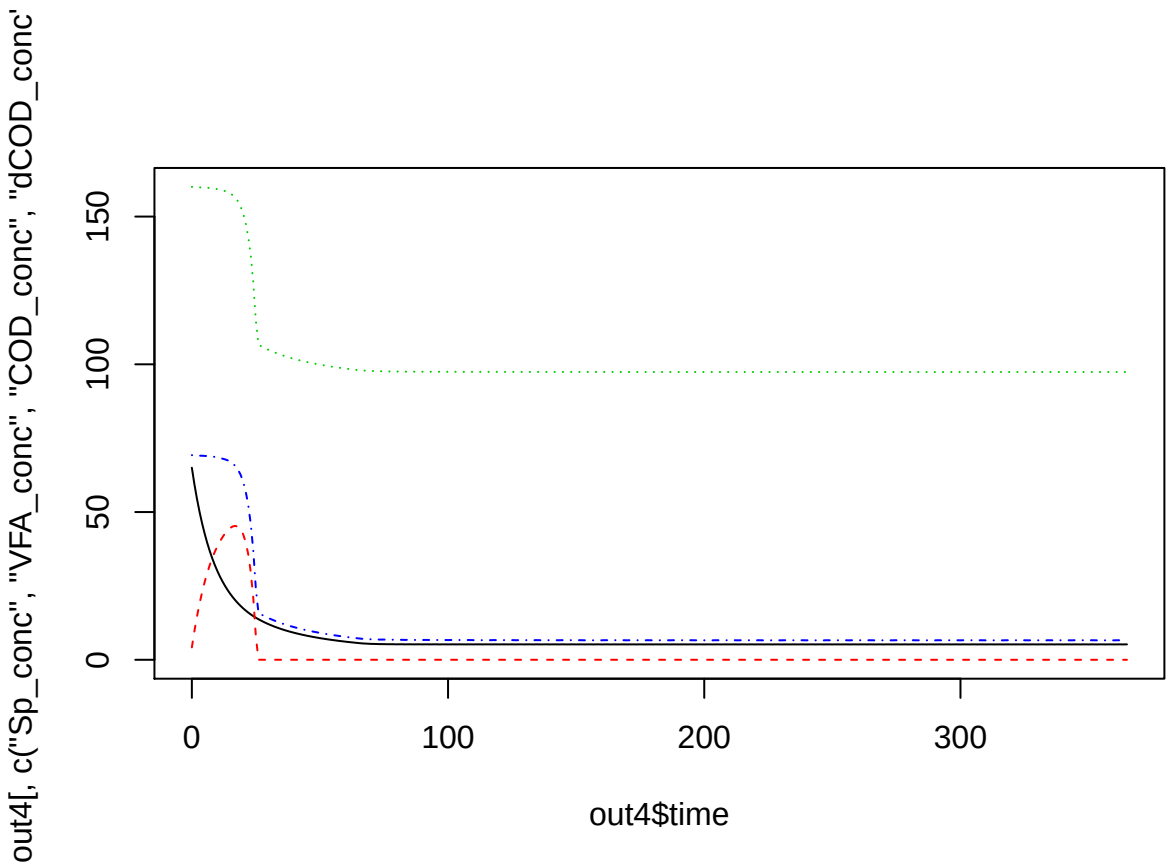

```
plot(f_COD_CH4_rate ~ time, data = out4, type = 'l')
```




```r
matplot(out4$time, out4[, nn <- c('m1_conc', 'm2_conc', 'm3_conc', 'p1_conc', 'p2_conc')], type = 'l')
legend('topleft', nn, col = 1:5, lty = 1)
```



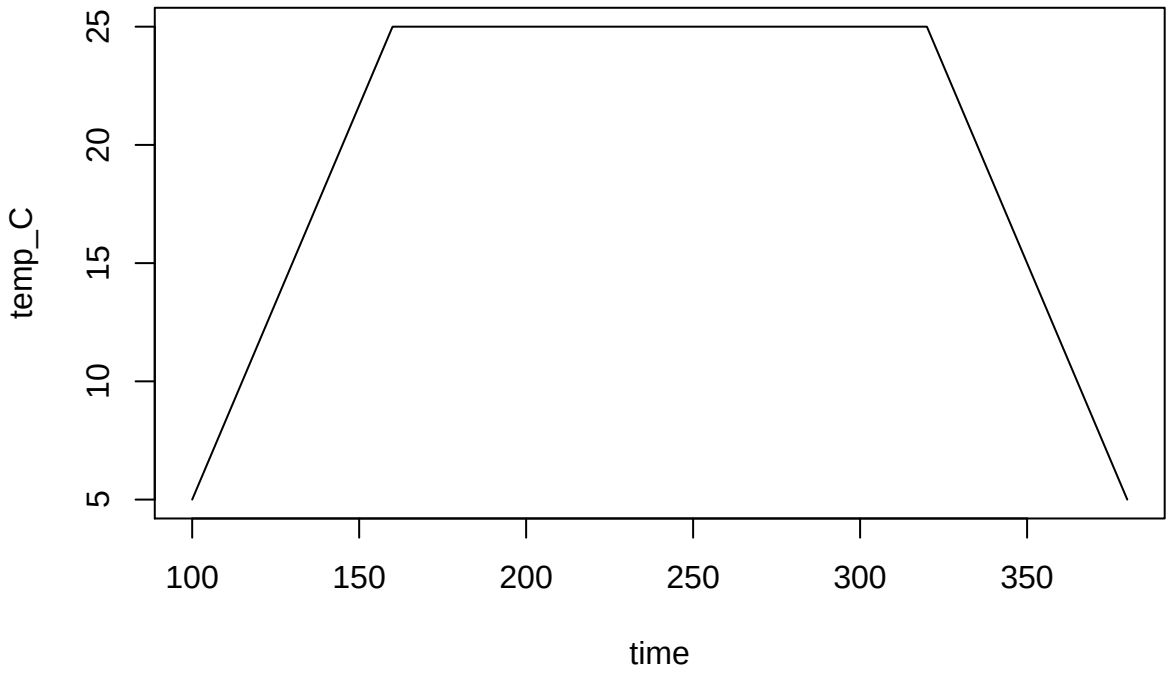
```
matplot(out4$time, out4[, c('Sp_conc', 'VFA_conc', 'COD_conc', 'dCOD_conc')], type = 'l')
```



Temperature variation

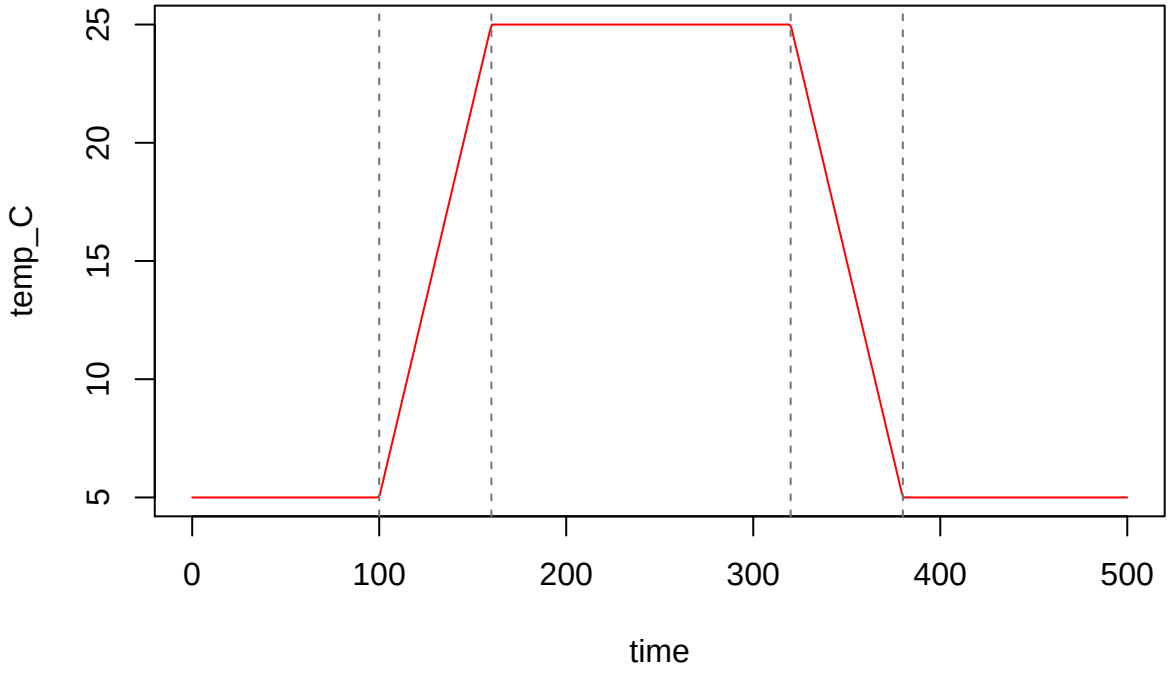
Predicting short- and long-term responses to temperature change was a central objective of the ATM99 model. Variable temperature is entered in a data frame with two columns. For example, gradual warming from 5°C to 25°C, a hold, and then a gradual cooling back to 5°C can be specified as shown in the `temp_dat` data frame constructed below.

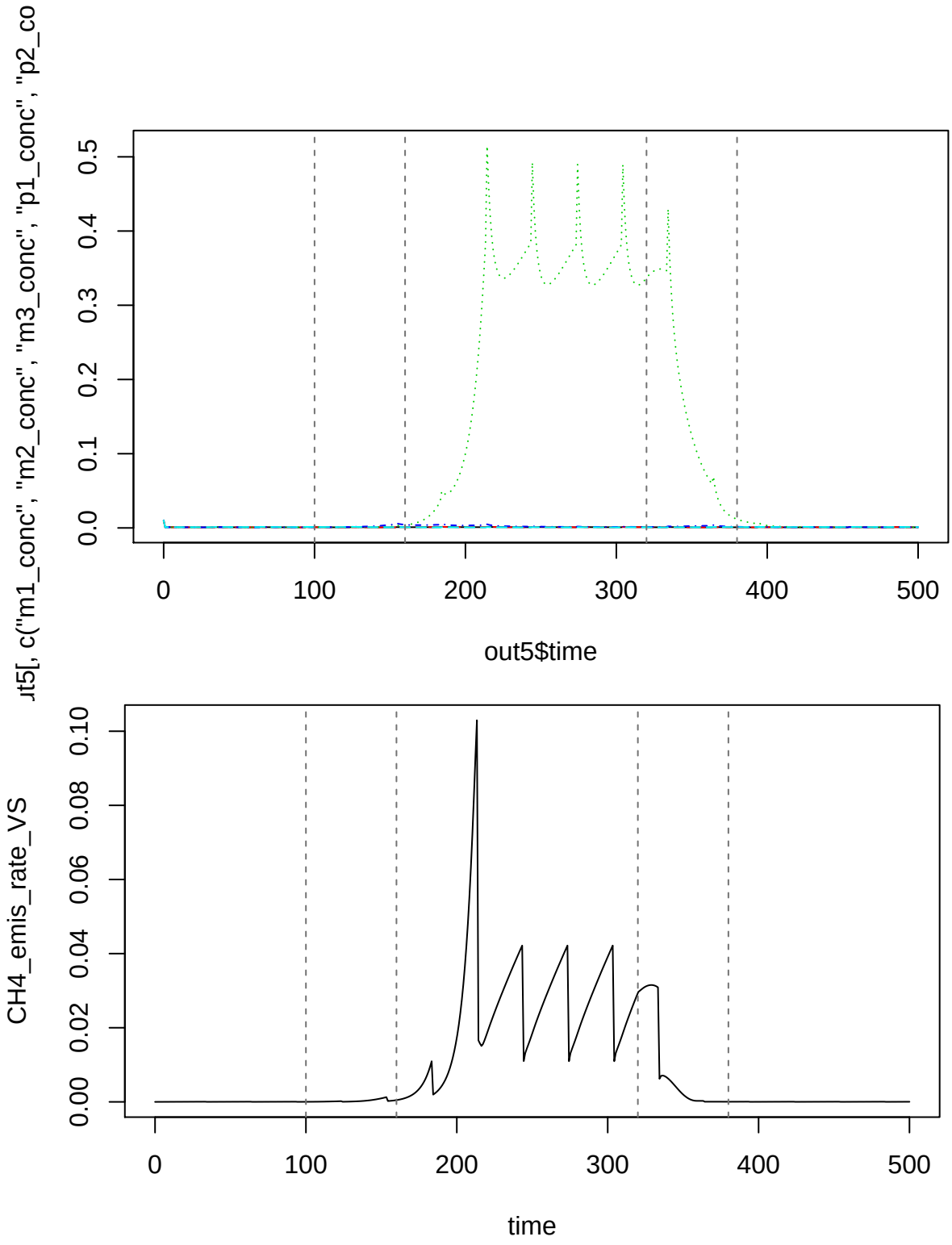
```
temp_dat <- data.frame(time = 100 + c(0, 60, 220, 280),
                       temp_C =     c(5, 25, 25,  5))
plot(temp_C ~ time, data = temp_dat, type = 'l')
```

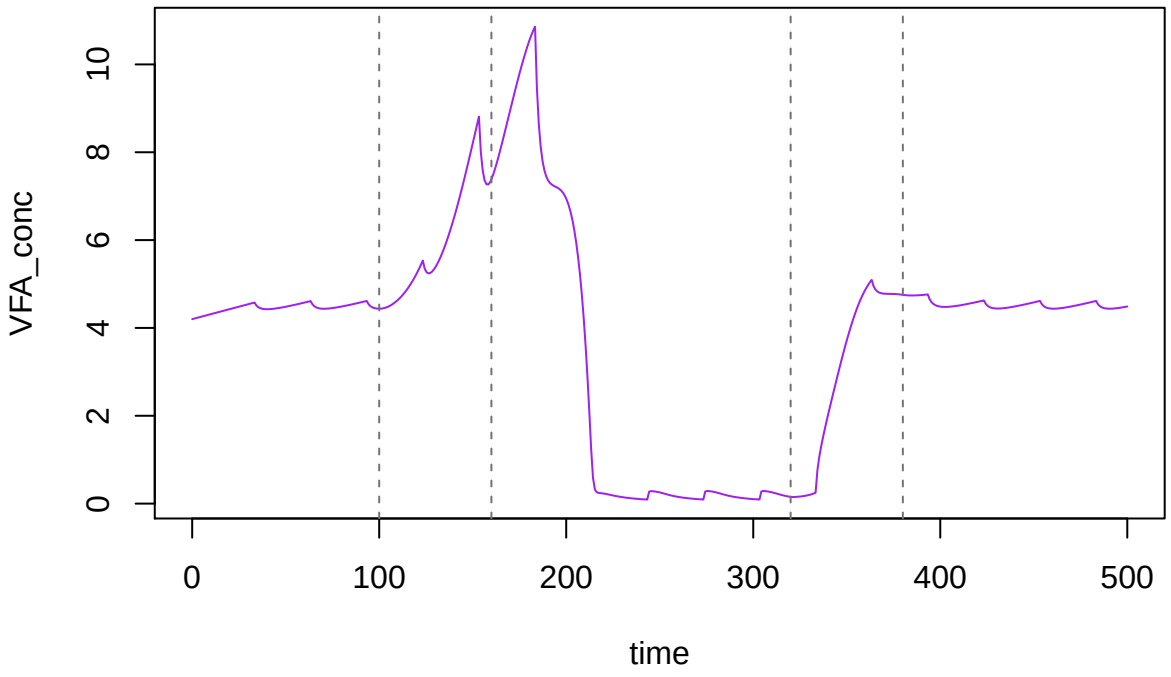


The model can either interpolate (the default) or use constant temperatures between change points. The temperature data can be supplied using the `mng_pars` argument or, more simply, with `add_pars`.

```
out5 <- atm(500, 1, add_pars = list(temp_C = temp_dat))
```




ut5[, c("m1_conc", "m2_conc", "m3_conc", "p1_conc", "p2_co
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
400
500
out5$time
CH4_emis_rate_VS
0.00
0.02
0.04
0.06
0.08
0.10
0
100
200
300
400
500
time


VFA_conc
0
2
4
6
8
10
0
100
200
300
400
500
time