Getting started with the biogas package

Charlotte Rennuit and Sasha D. Hafner (sasha.hafner@bce.au.dk)

April 5, 2025

1 Introduction

Anaerobic digestion is a popular technology for production of renewable energy and stabilisation of organic wastes, and research on the topic is carried out in laboratories in many countries. Transformation of raw data collected in laboratory experiments into quantities and rates of methane (CH_4) production requires a sequence of simple calculations. Although conceptually simple, these steps are time-consuming, and seldom described in detail in publications, so results may not be reproducible among laboratories or experiments. We developped the biogas package to address these issues. This document provides a brief introduction to the biogas package for new users. We have assumed that readers are familiar with biogas data collection and R.

2 Overview of functions

The package includes several "low-level" functions (Table 1) and "high-level" functions (Table 2). To go from data collected in the laboratory to biogas and methane (CH_4) production or biochemical methane potential (BMP), two high-level functions are needed: `cumBg()` (now replaced by `calcBg*()` functions) and `summBg()`. Comparing results to theory is facilitated by the remaining high-level function: `predBg()`. The low-level functions support the calculations carried out by the high-level functions, and may also be useful for some simple operations (e.g., converting reported biogas volumes to different standard conditions). This document describes the use of the high-level functions.

This vignette does not cover the latest developments in the package, but still provides a good overview. The newer `calcBg*()` functions are described in individual vignettes for the volumetric (`calcBgVol`) and manometric (`calcBgMan`) alternatives. The gravimetric version (`calcBgGrav`) is only described in its help file so far.

Table 1: Operations done with the low-level functions in the biogas package. All functions are vectorized. See help files for more details.

Operation	Function
Standardise gas volume	`stdVol()`
Interpolate composition etc.	`interp()`
Calculate oxygen demand of a compound	`calcCOD()`
Calculate molar mass of a compound	`molMass()`
Calculate biogas volume from mass loss	`mass2vol()`
Calculate mass loss from biogas volume	`vol2mass()`
Convert gas volume to moles	`vol2mol()`

Table 2: Operations done with the high-level functions in the biogas package. The `cumBg()` and `summBg()` functions can handle data from any number of bottles. `predBg()` is vectorized.

Operation	Function
Calculate cumulative CH_4 production and rates from volume (mass), composition	`cumBg()`
Calculate biochemical methane potential, summarise cumulative production or rates	`summBg()`
Predict biogas production based on substrate composition	`predBg()`

3 An example: calculation and prediction of biochemical methane potential

Calculation of biochemical methane potential (BMP) typically requires three data frames: initial mass, biogas quantity (volume, pressure, or bottle mass loss), and biogas composition. Input data may be structured in one of three ways: "long", "wide", or "combined". In a "long" format (`data.struct = 'long'`, the default), the measured variable (e.g., biogas volume) is in a single column (Fig. 1). In this case columns with unique bottle IDs and time allow the `biogas` functions to link observations in the two data frames[1]. Any order of observations can be used in input data frames.

The third data frame on initial conditions is used by the `summBg()` function. It should contain at least a bottle ID column and a description of the bottle contents. If the contribution of an inoculum is to be subtrated (as in the BMP test), the mass of inoculum added should be included here. Any measurements to be used to normalise biogas or CH_4 production are included here, using a "wide" format (Fig. 2). Note that there is no time column in this data frame– these values are independent of time.

[1] But observations need not be for the same times. Interpolation by `interp` takes care of this. Note that the time columns can be date/time objects as well as numeric or integer.

Reactor ID	Time	Response variable (volume or mass)
R1	1	$y_{1,1}$
R2	1	$y_{2,1}$
...	...	...
R_n	1	$y_{i,1}$
R1	2	$y_{1,2}$
R2	2	$y_{2,2}$
R_n	2	$y_{i,2}$
...	...	...
R_n	t_k	$y_{n,k}$

Reactor ID	Time	Response variable (Composition)
R1	2	$y_{1,2}$
R2	2	$y_{2,2}$
...	...	...
R_n	2	$y_{n,2}$
...	...	...
R_n	t_k	$y_{n,k}$

Figure 1: General structure of time-dependent data frames for the `dat` (left) and `comp` (right) arguments to the `cumBg()` function.

With the "wide" data structure (`data.struct = 'wide'`) the biogas quantity data frame contains a separate column for each bottle. And in the "combined" option (`data.struct = 'longcombo'`) a single data frame contains both biogas quantity and composition in a "long" structure.

Reactor ID	Description	Substrate VS mass	Inoculum total mass	...
R1	Substrate A	10.2	302	...
R2	Substrate A	9.85	301	...
R3	Substrate A	10.3	298	...
R4	Substrate B	8.5	300	...
...	...	...	...	...
R6	Inoculum only		502	...
...	...	...	...	...
R_n	...	...	...	...

Figure 2: General structure of initial conditions data frame for the `setup` argument to the `summBg()` function.

In this example, we will use the example data sets included with the package: `vol` for biogas volumes, `comp` for composition, and `setup` for grouping and substrate and inoculum masses. These data are from a BMP test that was carried out on two different substrates A and B, and cellulose (included as a "control"). The experiment included 12 batch bottles:

- 3 bottles with substrate A and inoculum
- 3 bottles with substrate B and inoculum
- 3 bottles with cellulose and inoculum
- 3 bottles with inoculum only

Reactors consisted of 500 mL or 1.0 L glass bottles, and were sealed with a butyl rubber septum and a screw cap. Initial substrate and inoculum masses were determined. A typical volumetric method was used to measure biogas production: accumulated biogas was measured and removed intermittently using syringes, and composition was measured for some of these samples.

```
library(biogas)

data("vol")

dim(vol)

## [1] 288   4

head(vol)

##    id           date.time days vol
## 1 2_1 2014-06-07 07:00:00 1.98 393
## 2 2_1 2014-06-08 07:00:00 2.98 260
## 3 2_1 2014-06-09 07:00:00 3.98 245
## 4 2_1 2014-06-10 07:00:00 4.98 225
## 5 2_1 2014-06-11 07:00:00 5.98 200
## 6 2_1 2014-06-12 08:00:00 7.02 175

summary(vol)

##        id        date.time                          days
##  2_1    : 24   Min.   :2014-06-07 07:00:00.0   Min.   :  1.98
##  2_2    : 24   1st Qu.:2014-06-13 20:00:00.0   1st Qu.:  8.52
##  2_3    : 24   Median :2014-06-28 06:00:00.0   Median : 22.94
##  2_4    : 24   Mean   :2014-07-16 15:29:22.5   Mean   : 41.33
##  2_5    : 24   3rd Qu.:2014-07-25 22:45:00.0   3rd Qu.: 50.63
##  2_6    : 24   Max.   :2014-12-19 04:30:00.0   Max.   :196.92
##  (Other):144
```

```
##       vol
##  Min.   : 98.0
##  1st Qu.:171.5
##  Median :225.0
##  Mean   :271.7
##  3rd Qu.:300.0
##  Max.   :840.0
##
```

```
data("comp")

dim(comp)

## [1] 132   4

head(comp)

##      id           date.time  days      xCH4
## 516 2_1 2014-06-12 08:00:00  7.02 0.7104731
## 519 2_1 2014-06-19 08:00:00 14.02 0.7024937
## 522 2_1 2014-06-26 05:00:00 20.90 0.6659919
## 524 2_1 2014-07-03 04:00:00 27.85 0.6789466
## 525 2_1 2014-07-10 03:00:00 34.81 0.6951429
## 528 2_1 2014-07-24 04:00:00 48.85 0.6693053

summary(comp)

##        id       date.time                           days
##  2_1    :11   Min.   :2014-06-12 08:00:00.00   Min.   :  7.02
##  2_2    :11   1st Qu.:2014-06-26 05:00:00.00   1st Qu.: 20.90
##  2_3    :11   Median :2014-07-24 04:00:00.00   Median : 48.85
##  2_4    :11   Mean   :2014-07-31 07:47:43.64   Mean   : 56.01
##  2_5    :11   3rd Qu.:2014-08-28 04:00:00.00   3rd Qu.: 83.85
##  2_6    :11   Max.   :2014-10-13 07:00:00.00   Max.   :129.98
##  (Other):66
##       xCH4
##  Min.   :0.5647
##  1st Qu.:0.6393
##  Median :0.6598
##  Mean   :0.6587
##  3rd Qu.:0.6786
##  Max.   :0.7115
##
```

```
data("setup")

setup
```

```
##      id descrip   msub  minoc   mvs.sub mvs.inoc  mcod.sub mcod.inoc
## 1   2_1       A 178.96 328.82  3.839567 12.92268  5.527522  19.09109
## 5   2_2       A 178.58 350.90  3.831414 13.79043  5.515785  20.37305
## 6   2_3       A 178.58 326.61  3.831414 12.83583  5.515785  18.96278
## 7   2_4       B  40.21 465.32  5.333816 18.28716  8.325115  27.01620
## 8   2_5       B  40.04 461.90  5.311266 18.15275  8.289918  26.81764
## 9   2_6       B  40.13 475.61  5.323204 18.69156  8.308551  27.61363
## 10  2_7   cellu   5.75 500.94  5.507470 19.68703  7.762500  29.08428
## 11  2_8   cellu   5.76 498.10  5.517048 19.57542  7.776000  28.91939
## 12  2_9   cellu   5.71 504.65  5.469157 19.83283  7.708500  29.29968
## 2  2_10    inoc 501.50 501.50 19.709037 19.70904 29.116792  29.11679
## 3  2_11    inoc 502.27 502.27 19.739298 19.73930 29.161498  29.16150
## 4  2_12    inoc 502.12 502.12 19.733403 19.73340 29.152789  29.15279
##     m.tot  mvs.tot mcod.tot
## 1  657.78 16.76225 24.61862
## 5  679.79 17.62184 25.88883
## 6  654.68 16.66724 24.47857
## 7  655.22 23.62097 35.34132
## 8  652.56 23.46402 35.10756
## 9  665.76 24.01476 35.92219
## 10 656.68 25.19450 36.84678
## 11 653.02 25.09246 36.69539
## 12 659.28 25.30199 37.00818
## 2  652.07 19.70904 29.11679
## 3  752.37 19.73930 29.16150
## 4  650.66 19.73340 29.15279
```

3.1 Cumulative production

The first step in processing these data is to calculate cumulative production of biogas and CH_4 and production rates. We can do this with the `cumBg()` function, using `vol` and `comp` data frames as input. The arguments for the function are:

```
  args(cumBg)
```

```
## function (dat, dat.type = "vol", comp = NULL, temp = NULL, pres = NULL,
##     interval = TRUE, data.struct = "long", id.name = "id", time.name = "time",
##     dat.name = dat.type, comp.name = "xCH4", pres.resid = NULL,
##     temp.init = NULL, pres.init = NULL, rh.resid = NULL, rh.resid.init = 1,
##     headspace = NULL, vol.hs.name = "vol.hs", headcomp = "N2",
```

```
##     absolute = TRUE, pres.amb = NULL, mol.f.name = NULL, vol.syr = NULL,
##     cmethod = "removed", imethod = "linear", extrap = FALSE,
##     addt0 = TRUE, showt0 = TRUE, dry = FALSE, empty.name = NULL,
##     std.message = !quiet, check = TRUE, temp.std = getOption("temp.std",
##         as.numeric(NA)), pres.std = getOption("pres.std", as.numeric(NA)),
##     unit.temp = getOption("unit.temp", "C"), unit.pres = getOption("unit.pres",
##         "atm"), quiet = FALSE)
## NULL
```

Most of the arguments have default values, but to calculate CH_4 production we must provide values for at least `dat` (we will use `vol`), `comp` (we will use `comp`), `temp` (biogas temperature), and `pres` (biogas pressure)[2], along with the names of a few columns in our input data frames. We need to specify the name of the time column in `vol` and `comp` using the `time.name` argument. This name must be the same in both data frames. Similarly, there is an `id.name` argument for the bottle ID column (used to match up volume and composition data), but we can use the default value (`"id"`) here because it matches the column name in `vol` and `comp`. And, the `comp.name` argument is used to indicate which column within the `comp` data frame contains the CH_4 content (as mole fraction in dry biogas, normalised so the sum of mole fractions of CH_4 and CO_2 sum to unity). We can use the default (`"xCH4"`) because it matches the name in `comp`. Lastly, the name of the column that contains the response variable in the `dat` data frame (`vol` here) can be specified with the `dat.name` argument. Here too we can use the default (`"vol"` for volumetric measurements or `"mass"` for gravimetric). By default (`cmethod = "removed"`) the function calculates volumes following [2] as the product of standardised volume of biogas removed and normalised CH_4 content.

```
cum.prod <- cumBg(vol, comp = comp, time.name = "days", temp = 35, pres = 1,
                  extrap = TRUE)

## Biogas composition is interpolated.
## Working with volume data, applying volumetric method.
## Using a standard pressure of 1 atm and standard temperature of 0
C for standardizing volume.
```

Note the message about standard temperature and pressure–it is important to make sure these values are correct, therefore users are reminded by a message[3]. The output looks like this:

[2] By default, temperature is in °C and pressure in atm, but these can be changed in the function call with the `temp.unit` and `pres.unit` arguments, or globally with `options`.

[3] Remember that standard conditions can be set in the function call with `temp.std` and `pres.std`, or globally with `options()`.

```
head(cum.prod)
```

```
##    id           date.time days vol      xCH4 temperature pressure      vBg
## 1 2_1                <NA> 0.00  NA        NA          NA       NA   0.0000
## 2 2_1 2014-06-07 07:00:00 1.98 393 0.7104731          35        1 328.9470
## 3 2_1 2014-06-08 07:00:00 2.98 260 0.7104731          35        1 217.6240
## 4 2_1 2014-06-09 07:00:00 3.98 245 0.7104731          35        1 205.0687
## 5 2_1 2014-06-10 07:00:00 4.98 225 0.7104731          35        1 188.3284
## 6 2_1 2014-06-11 07:00:00 5.98 200 0.7104731          35        1 167.4031
##       vCH4      cvBg    cvCH4     rvBg    rvCH4
## 1   0.0000    0.0000   0.0000       NA       NA
## 2 233.7080  328.9470 233.7080 166.1348 118.0343
## 3 154.6160  546.5710 388.3240 217.6240 154.6160
## 4 145.6958  751.6397 534.0198 205.0687 145.6958
## 5 133.8023  939.9681 667.8221 188.3284 133.8023
## 6 118.9354 1107.3712 786.7574 167.4031 118.9354
```

```
dim(cum.prod)
```

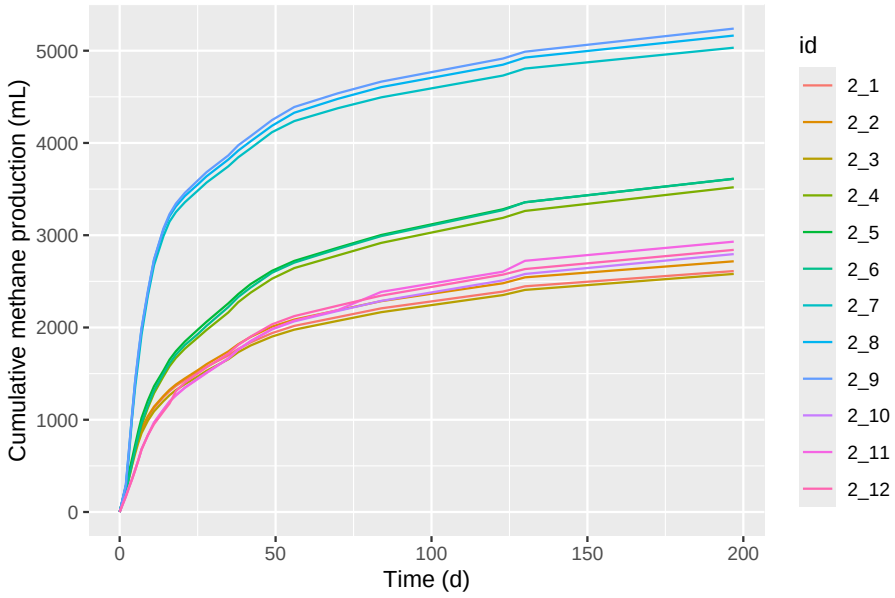
```
## [1] 300  13
```

The data frame that is returned has all the original columns in `vol`, plus others. In these columns, `v` stands for (standardised) volume, `cv` (standardised) cumulative volume, `rv` stands for (standardised) volume production rate, and `Bg` and `CH4` for biogas and methane. So `cvCH4` contains standardised cumulative CH_4 production. It is probably easier to understand the data in the output graphically. Here we'll use the `qplot` function from the `ggplot2` package to plot it.

```
library(ggplot2)

qplot(x = days, y = cvCH4, data = cum.prod, xlab = "Time (d)",
      ylab = "Cumulative methane production (mL)", color = id,
      geom = "line")
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning
was
## generated.
```



3.2 Other data structures

As of biogas version 1.5.0, the "long" data structures described above is not the only option. In addition, "wide" and combined "long" structures are possible. We can compare the three possible approaches using the same dataset.

Let's load data on biogas production from three bottles with wastewater sludge.

```
data("s3voll")
data("s3compl")
data("s3volw")
data("s3compw")
data("s3lcombo")
```

The "long" structure described above looks like this:

```
s3voll
```

```
##    id  time.d vol.ml cvol.ml
## 1   D  0.9438    103     103
## 2   E  0.9451    106     106
## 3   F  0.9472    107     107
## 4   D  2.9060    192     295
## 5   E  2.9090    181     287
## 6   F  2.9100    203     310
## 7   D  5.8860    141     436
```

```
## 8   E  5.8880    133     420
## 9   F  5.8900    140     450
## 10  D 10.0000    112     548
## 11  E 10.0000    111     531
## 12  F 10.0100    110     560
## 13  D 23.1000    200     748
## 14  E 23.1000    190     721
## 15  F 23.1000    200     760
## 16  D 34.0100    109     857
## 17  E 34.0100    110     831
## 18  F 34.0100    112     872
## 19  D 57.8400    146    1003
## 20  E 57.8400    136     967
## 21  F 57.8400    138    1010

s3compl

##    id time.d   xCH4
## 1   D  2.906 0.6983
## 2   E  2.909 0.6817
## 3   F  2.910 0.6869
## 4   D 10.000 0.6646
## 5   E 10.000 0.6644
## 6   F 10.010 0.6632
## 7   D 23.100 0.6946
## 8   E 23.100 0.6871
## 9   F 23.100 0.6829
## 10  D 34.010 0.6626
## 11  E 34.010 0.6556
## 12  F 34.010 0.6527
## 13  D 57.840 0.6651
## 14  E 57.840 0.6600
```

The "wide" format contains (mostly) the same data, but there are separate columns for each bottle.

```
s3volw

##    time.d   D   E   F
## 1  0.9438 103 106 107
## 2  2.9060 192 181 203
## 3 34.0100 109 110 112
## 4  5.8860 141 133 140
## 5 10.0000 112 111 110
## 6 23.1000 200 190 200
## 7 57.8400 146 136 138
```

```
s3compw
```

```
##   time.d      D      E      F
## 1  2.906 0.6983 0.6817 0.6869
## 2 34.010 0.6626 0.6556 0.6527
## 3 10.000 0.6646 0.6644 0.6632
## 4 23.100 0.6946 0.6871 0.6829
## 5 57.840 0.6651 0.6600     NA
```

Note the missing composition value in `s3compw`. With the "long" structure, a row was simply omitted. Both approaches will result in the same output though. With the "wide" approach all bottles must be measured at the same times.

Finally, in the combined approach both volume and composition are in the same "long" data frame.

```
s3lcombo
```

```
##    id  time.d vol.ml   xCH4
## 1   D  0.9438    103     NA
## 2   E  0.9451    106     NA
## 3   F  0.9472    107     NA
## 4   D  2.9060    192 0.6983
## 5   E  2.9090    181 0.6817
## 6   F  2.9100    203 0.6869
## 7   D  5.8860    141 0.6800
## 8   E  5.8880    133 0.6800
## 9   F  5.8900    140 0.6800
## 10  D 10.0000    112 0.6646
## 11  E 10.0000    111 0.6644
## 12  F 10.0100    110 0.6632
## 13  D 23.1000    200 0.6946
## 14  E 23.1000    190 0.6871
## 15  F 23.1000    200 0.6829
## 16  D 34.0100    109 0.6626
## 17  E 34.0100    110 0.6556
## 18  F 34.0100    112 0.6527
## 19  D 57.8400    146 0.6651
## 20  E 57.8400    136 0.6600
## 21  F 57.8400    138     NA
```

Each of these structures can be used by `cumBg` by changing the `comp` argument.

```
cpl <- cumBg(s3lcombo, comp = s3compl, temp = 25, pres = 1,
             id.name = 'id', time.name = 'time.d',
```

```
             dat.name = 'vol.ml', comp.name = 'xCH4',
             extrap = TRUE)

## Biogas composition is interpolated.
## Working with volume data, applying volumetric method.
## Using a standard pressure of 1 atm and standard temperature of 0
C for standardizing volume.

cpw <- cumBg(s3volw, comp = s3compw, temp = 25, pres = 1,
             time.name = 'time.d',
             data.struct = 'wide',
             dat.name = 'D', comp.name = 'D',
             extrap = TRUE)

## Biogas composition is interpolated.
## Working with volume data, applying volumetric method.
## Using a standard pressure of 1 atm and standard temperature of 0
C for standardizing volume.

cpc <- cumBg(s3lcombo, temp = 25, pres = 1,
             id.name = 'id', time.name = 'time.d',
             data.struct = 'longcombo',
             dat.name = 'vol.ml', comp.name = 'xCH4',
             extrap = TRUE)

## Biogas composition is interpolated.
## Working with volume data, applying volumetric method.
## Using a standard pressure of 1 atm and standard temperature of 0
C for standardizing volume.
```

Output is nearly identical here. The small differences result from the use of unique times for each bottle in the long formats.

```
head(cpl)

##   id  time.d vol.ml      xCH4 temperature pressure       vBg      vCH4
## 1  D  0.0000     NA        NA          NA       NA   0.00000   0.00000
## 2  D  0.9438    103 0.6983000          25        1  91.40334  63.82696
## 3  D  2.9060    192 0.6983000          25        1 170.38293 118.97840
## 4  D  5.8860    141 0.6841435          25        1 125.12497  85.60344
## 5  D 10.0000    112 0.6646000          25        1  99.39004  66.05462
## 6  D 23.1000    200 0.6946000          25        1 177.48222 123.27915
##        cvBg     cvCH4     rvBg     rvCH4
## 1   0.00000   0.00000       NA        NA
## 2  91.40334  63.82696 96.84609 67.627628
## 3 261.78628 182.80536 86.83260 60.635206
## 4 386.91124 268.40879 41.98824 28.725985
```

```
## 5 486.30129 334.46342 24.15898 16.056058
## 6 663.78351 457.74257 13.54826  9.410622

head(cpw)

##   id  time.d vol      xCH4 temperature pressure       vBg      vCH4
## 1  D  0.0000  NA        NA          NA       NA   0.00000   0.00000
## 2  D  0.9438 103 0.6983000          25        1  91.40334  63.82696
## 3  D  2.9060 192 0.6983000          25        1 170.38293 118.97840
## 4  D  5.8860 141 0.6841435          25        1 125.12497  85.60344
## 5  D 10.0000 112 0.6646000          25        1  99.39004  66.05462
## 6  D 23.1000 200 0.6946000          25        1 177.48222 123.27915
##        cvBg     cvCH4     rvBg     rvCH4
## 1   0.00000   0.00000       NA        NA
## 2  91.40334  63.82696 96.84609 67.627628
## 3 261.78628 182.80536 86.83260 60.635206
## 4 386.91124 268.40879 41.98824 28.725985
## 5 486.30129 334.46342 24.15898 16.056058
## 6 663.78351 457.74257 13.54826  9.410622

head(cpc)

##   id  time.d vol.ml   xCH4 temperature pressure       vBg      vCH4
## 1  D  0.0000     NA     NA          NA       NA   0.00000   0.00000
## 2  D  0.9438    103 0.6983          25        1  91.40334  63.82696
## 3  D  2.9060    192 0.6983          25        1 170.38293 118.97840
## 4  D  5.8860    141 0.6800          25        1 125.12497  85.08498
## 5  D 10.0000    112 0.6646          25        1  99.39004  66.05462
## 6  D 23.1000    200 0.6946          25        1 177.48222 123.27915
##        cvBg     cvCH4     rvBg     rvCH4
## 1   0.00000   0.00000       NA        NA
## 2  91.40334  63.82696 96.84609 67.627628
## 3 261.78628 182.80536 86.83260 60.635206
## 4 386.91124 267.89033 41.98824 28.552006
## 5 486.30129 333.94496 24.15898 16.056058
## 6 663.78351 457.22411 13.54826  9.410622
```

3.3 Calculating BMP from cumulative production

To calculate BMP we need to substract the contribution of the inoculum to CH_4 production for each bottle, normalise by substrate volatile solids (VS), and calculate means and standard deviations. This is done by the `summBg()` function using the results from `cumBg()`, along with the `setup` data frame. The arguments for `summBg()` are:

```
  args(summBg)
```

```
## function (vol, setup, id.name = "id", time.name = "time", descrip.name = "descrip",
##     inoc.name = NULL, inoc.m.name = NULL, norm.name = NULL, norm.se.name = NULL,
##     vol.name = "cvCH4", imethod = "linear", extrap = FALSE, when = 30,
##     when.min = 0, rate.crit = "net", show.obs = FALSE, show.rates = FALSE,
##     show.more = FALSE, sort = TRUE, set.name = "set", quiet = FALSE)
## NULL
```

This is a flexible function, and is useful for more than just calculating BMP. For example, to simply determine the mean cumulative CH_4 production for each substrate at 30 d, we could use:

```
summBg(cum.prod, setup = setup, time.name = "days", descrip.name = "descrip",
       when = 30)
```

```
## Response variable (volume) is cum.prod$cvCH4.
## Inoculum contribution not subtracted.
## No normalization by substrate mass.
```

```
##   descrip days     mean       se       sd n
## 1       A   30 1608.362 22.06860 38.22393 3
## 2       B   30 2078.248 26.48733 45.87740 3
## 3   cellu   30 3686.127 33.35392 57.77069 3
## 4    inoc   30 1575.326 19.18801 33.23460 3
```

Here, the response variable was `cvCH4` (cumulative CH_4 production, the default–but `vol.name` could be used to specify any column). The argument `descrip.name` is the name of the column in `setup` that gives a description of the bottle. Here it is used for grouping bottles. We could have used the default value in this call.

To calculate BMP, we need to provide information on where inoculum and substrate VS masses can be found. To subtract the inoculum contribution, we need to provide a value for the `inoc.name` argument, which should be the value in the `setup$descrip.name` column that indicates that the bottle contained inoculum only. In our `setup` data frame, the value is `"inoc"`. Inoculum mass is given in the `minoc` column, and we need to provide this information using the `inoc.m.name` argument (although here also, we could use the default value). The last step is normalisation of cumulative CH_4 production, based on substrate VS mass. This mass must be stored in the `setup` data frame and the name of column is given using the `norm.name` argument. Here, it is `"mvs.sub"`. We will evaluate CH_4 production at at time selection by the function when relative methane production drops below 1% of cumulative (after subtracting inoculum production) per day for at least 3 days (`when` argument).

```
BMP <- summBg(cum.prod, setup = setup, time.name = "days", inoc.name = "inoc",
              inoc.m.name = "minoc", norm.name = "mvs.sub", when = "1p3d")

## Response variable (volume) is cum.prod$cvCH4.
## Inoculum contribution subtracted based on setup$minoc.
## Response normalized by setup$mvs.sub.

BMP

##   descrip  days     mean       se        sd n rate.crit.met
## 1       A 34.81 149.6533 4.146875  7.182599 3          TRUE
## 2       B 42.00 128.9929 6.916043 11.978937 3          TRUE
## 3   cellu 20.90 372.8132 6.598490 11.428920 3          TRUE
```

Note the messages–because any response variable could be used and subtraction of an inoculum contribution and normalisation are optional, it is important to check these messages and be sure that `summBg()` did what you think it did. Additionally, it is good practice to view and save results from individual bottles, and check the apparent contribution of the inoculum to each bottle's biogas production. This additional information can be returned by setting `show.obs = TRUE`.

3.4 Predicting methane production

The function `predBg()` provides a flexible approach for predicting methane potential, and in our example can be used to quickly check our experimental values. Predictions can be based on an empirical chemical formula, chemical oxygen demand (COD), or macromolecule composition.

Our BMP test included cellulose as a control. Using its chemical formula ($C_6H_10O_5$), we can calculate theoretical methane potential to compare to our measurements[4].

```
predBg("C6H10O5")

## [1] 413.7274
```

So we see that theoretical methane potential of cellulose is 414 mL g^{-1}. Comparing expected cellulose BMP to measurements is an important way to check BMP experiments. How does this compare to our measurements?

```
BMP

##   descrip  days     mean       se        sd n rate.crit.met
## 1       A 34.81 149.6533 4.146875  7.182599 3          TRUE
## 2       B 42.00 128.9929 6.916043 11.978937 3          TRUE
## 3   cellu 20.90 372.8132 6.598490 11.428920 3          TRUE
```

[4] In this case, the calculation is based on Eq. (13.5) in Rittmann and McCarty [3]. When the input is COD, it is based on the COD of CH_4, as described in [3].

The measured value is a bit lower, which is reasonable. It is common to assume that $5 - 10\%$ of substrate is used to produce microbial biomass, and so not converted to biogas. We can incorporate this assumption into our prediction using the `fs` argument, which is the fraction of substrate electrons used for cell synthesis.

```
predBg("C6H10O5", fs = 0.1)
```

```
## [1] 372.3547
```

Measured and predicted values are close after making this correction.

We don't have empirical formulas for substrates A and B, but we can predict theoretical potential by using the COD. Initial COD masses are in the `setup` data frame, and from these we can calculate COD:VS ratios for substrates A and B of 1.439 and 1.561 g g^{-1}. Cellulose has a calculated oxygen demand (COD′)[5] of 1.184 g g^{-1}. Predicted CH_4 production per g VS is therefore:

```
predBg(COD = c(A = 1.439, B = 1.561, cellu = 1.184))
```

```
## [1] 502.7638 545.3887 413.6709
```

Measured BMP was substantially lower for substrates A and B, indicating very low degradability. In fact, we could use `predBg()` to estimate effective degradability (ignoring synthesis of microbial biomass).

```
  BMP$mean/predBg(COD = c(A = 1.439, B = 1.561, cellu = 1.184))
```

```
## [1] 0.2976613 0.2365155 0.9012315
```

We see that substrates A and B had low degradability, while degradability of cellulose was high. Both substrates A and B were digestate from digesters, i.e., they had already been anaerobically digested once before these measurements, and so we should expect low degradability.

4 Continuing with the biogas package

The three functions demonstrated in this document can be used in other ways not described here. For example, `cumBg()` can be used with measurements of bottle mass over time to determine biogas production[1], `summBg()` can return results for multiple times, and `predBg()` function can predict microbial nitrogen requirements and biogas composition. More details can be found in the help files for these functions, or, for `predBg`, in the `predBg` vignette. The low-level functions are straight-forward to use, and details can also be found in the help files.

[5] Oxygen demand can be calculated with the `calcCOD` function.

To receive updates on the biogas package, you can subscribe to a mailing list by sending an e-mail to either of us. And please send us a message if you find a bug or have a suggestion for improving an existing function or adding a new one.

References

[1] S.D. Hafner, C. Rennuit, J.M. Triolo, and B.K. Richards. Validation of a simple gravimetric method for measuring biogas production in laboratory experiments. *Biomass and Bioenergy*, 83:297–301, 2015.

[2] B.K. Richards, R.J. Cummings, T.E. White, and W.J. Jewell. Methods for kinetic-analysis of methane fermentation in high solids biomass digesters. *Biomass & Bioenergy*, 1(2):65–73, 1991.

[3] B. E. Rittmann and P. L. McCarty. *Environmental Biotechnology: Principles and Applications*. McGraw-Hill series in water resources and environmental engineering. McGraw-Hill, Boston, 2001.