

# Getting started with the biogas package

Charlotte Rennuit ([cre@kbm.sdu.dk](mailto:cre@kbm.sdu.dk)) and Sasha D. Hafner ([saha@kbm.sdu.dk](mailto:saha@kbm.sdu.dk))

July 29, 2015

## 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 ( $\text{CH}_4$ ) 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 developed 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 six “low-level” functions (Table 1) and three “high-level” functions (Table 2). To go from data collected in the laboratory to biogas and methane ( $\text{CH}_4$ ) production or biochemical methane potential (BMP), two high-level functions are needed: `cumBg()` 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.

Table 1: Operations done with the low-level functions in the biogas package. All functions except `calcCOD()` and `molMass()` are vectorized. See help files for more details.

Operation	Function
Standardise gas volume	<code>stdVol()</code>
Interpolate composition etc.	<code>interp()</code>
Calculate oxygen demand of a compound	<code>calcCOD()</code>
Calculate molar mass of a compound	<code>molMass()</code>
Calculate biogas volume from mass loss	<code>mass2vol()</code>
Calculate mass loss from biogas volume	<code>vol2mass()</code>

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

Operation	Function
Calculate cumulative $\text{CH}_4$ production and rates from volume (mass), composition	<code>cumBg()</code>
Calculate biochemical methane potential, summarise cumulative production or rates	<code>summBg()</code>
Predict biogas production based on substrate composition	<code>predBg()</code>

### 3 An example: calculation and prediction of biochemical methane potential

Calculation of biochemical methane potential (BMP) typically requires three data frames: initial mass, biogas volume (or reactor mass loss), and biogas composition. Volume and composition data frames must be in a “long” format, with the measured variable in a single column (Fig. 1). Columns with unique reactor IDs and time allow the `biogas` functions to link observations in the two data frames<sup>1</sup>. Any order of observations can be used in input data frames. It is possible to put volume and composition data in a single data frame, but since composition is not always measured concurrently with volume, a better approach is to use two separate data frames.

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.

The third data frame on initial conditions is used by the `summBg()` function. It should contain at least a reactor ID column and a description of the reactor 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  $\text{CH}_4$  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.

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.

<sup>1</sup> 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.

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 reactors:

- 3 reactors with substrate A and inoculum
- 3 reactors with substrate B and inoculum
- 3 reactors with cellulose and inoculum
- 3 reactors 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 13:00:00 1.98 393
## 2 2_1 2014-06-08 13:00:00 2.98 260
## 3 2_1 2014-06-09 13:00:00 3.98 245
## 4 2_1 2014-06-10 13:00:00 4.98 225
## 5 2_1 2014-06-11 13:00:00 5.98 200
## 6 2_1 2014-06-12 14:00:00 7.02 175
summary(vol)
##      id      date.time      days
## 2_1      : 24   Min.   :2014-06-07 13:00:00   Min.   : 1.98
## 2_2      : 24   1st Qu.:2014-06-14 02:00:00   1st Qu.: 8.52
## 2_3      : 24   Median :2014-06-28 12:00:00   Median :22.94
## 2_4      : 24   Mean    :2014-07-16 21:29:22   Mean    :41.33
## 2_5      : 24   3rd Qu.:2014-07-26 04:45:00   3rd Qu.:50.63
## 2_6      : 24   Max.    :2014-12-19 10:30:00   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 14:00:00 7.02 0.7104731
```

```
## 519 2_1 2014-06-19 14:00:00 14.02 0.7024937
## 522 2_1 2014-06-26 11:00:00 20.90 0.6659919
## 524 2_1 2014-07-03 10:00:00 27.85 0.6789466
## 525 2_1 2014-07-10 09:00:00 34.81 0.6951429
## 528 2_1 2014-07-24 10:00:00 48.85 0.6693053
summary(comp)
##      id      date.time      days
## 2_1    :11    Min.    :2014-06-12 14:00:00    Min.    : 7.02
## 2_2    :11    1st Qu.:2014-06-26 11:00:00    1st Qu.: 20.90
## 2_3    :11    Median :2014-07-24 10:00:00    Median : 48.85
## 2_4    :11    Mean    :2014-07-31 13:47:43    Mean    : 56.01
## 2_5    :11    3rd Qu.:2014-08-28 10:00:00    3rd Qu.: 83.85
## 2_6    :11    Max.    :2014-10-13 13:00:00    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
## 1  2_1      A 178.96 328.82  3.839567 12.92268  5.527522
## 5  2_2      A 178.58 350.90  3.831414 13.79043  5.515785
## 6  2_3      A 178.58 326.61  3.831414 12.83583  5.515785
## 7  2_4      B  40.21 465.32  5.333816 18.28716  8.325115
## 8  2_5      B  40.04 461.90  5.311266 18.15275  8.289918
## 9  2_6      B  40.13 475.61  5.323204 18.69156  8.308551
## 10 2_7  cellu   5.75 500.94  5.507470 19.68703  7.762500
## 11 2_8  cellu   5.76 498.10  5.517048 19.57542  7.776000
## 12 2_9  cellu   5.71 504.65  5.469157 19.83283  7.708500
## 2  2_10  inoc 501.50 501.50 19.709037 19.70904 29.116792
## 3  2_11  inoc 502.27 502.27 19.739298 19.73930 29.161498
## 4  2_12  inoc 502.12 502.12 19.733403 19.73340 29.152789
##      mcod.inoc  m.tot  mvs.tot mcod.tot
## 1  19.09109 657.78 16.76225 24.61862
## 5  20.37305 679.79 17.62184 25.88883
## 6  18.96278 654.68 16.66724 24.47857
## 7  27.01620 655.22 23.62097 35.34132
## 8  26.81764 652.56 23.46402 35.10756
## 9  27.61363 665.76 24.01476 35.92219
## 10 29.08428 656.68 25.19450 36.84678
## 11 28.91939 653.02 25.09246 36.69539
## 12 29.29968 659.28 25.30199 37.00818
## 2  29.11679 652.07 19.70904 29.11679
## 3  29.16150 752.37 19.73930 29.16150
## 4  29.15279 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<sub>4</sub> 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, id.name = "id", time.name = "time",
##      dat.name = dat.type, comp.name = "xCH4", headspace = NULL,
##      vol.hs.name = "vol.hs", headcomp = "N2", temp.init = NULL,
##      temp.std = getOption("temp.std", 0),
##      pres.std = getOption("pres.std", 1),
##      unit.temp = getOption("unit.temp", "C"),
##      unit.pres = getOption("unit.pres", "atm"),
##      cmethod = "removed", imethod = "linear", extrap = FALSE,
##      addt0 = TRUE, showt0 = TRUE, std.message = TRUE,
##      check = TRUE)
## NULL
```

Most of the arguments have default values, but to calculate CH<sub>4</sub> 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)<sup>2</sup>, 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 reactor 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<sub>4</sub> content (as mole fraction in dry biogas, normalised so the sum of mole fractions of CH<sub>4</sub> and CO<sub>2</sub> 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<sub>4</sub> 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<sup>3</sup>. The output looks like this:

```
head(cum.prod)
##      id      date.time days vol      xCH4      vBg      vCH4
## 1 2_1      <NA> 0.00  0      NA      0.0000  0.0000
## 2 2_1 2014-06-07 13:00:00 1.98 393 0.7104731 328.9808 234.0289
## 3 2_1 2014-06-08 13:00:00 2.98 260 0.7104731 217.6463 154.8283
```

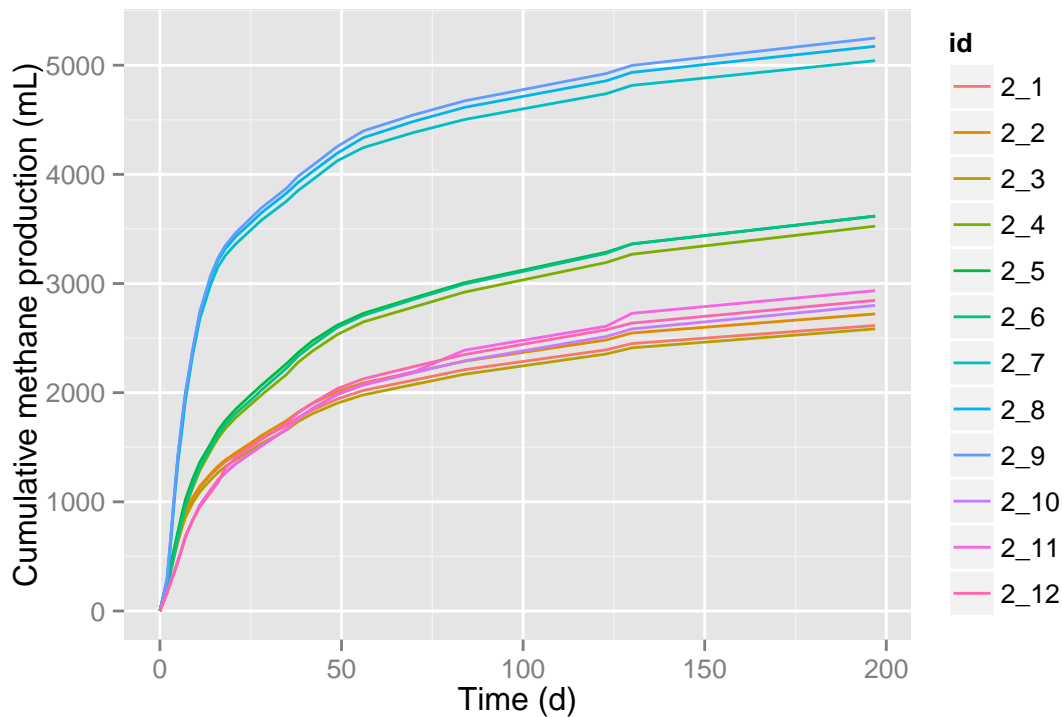
<sup>2</sup> By default, temperature is in °C and pressure in atm, but these can be changed in the function call with the `temp.unit` and `pres.unit` arguments, or globally with `options`.

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

```
## 4 2_1 2014-06-09 13:00:00 3.98 245 0.7104731 205.0898 145.8959
## 5 2_1 2014-06-10 13:00:00 4.98 225 0.7104731 188.3478 133.9860
## 6 2_1 2014-06-11 13:00:00 5.98 200 0.7104731 167.4202 119.0987
##      cvBg      cvCH4      rvBg      rvCH4
## 1      0.0000      0.0000      NA      NA
## 2 328.9808 234.0289 166.1519 118.1964
## 3 546.6271 388.8572 217.6463 154.8283
## 4 751.7168 534.7531 205.0898 145.8959
## 5 940.0646 668.7392 188.3478 133.9860
## 6 1107.4848 787.8379 167.4202 119.0987
dim(cum.prod)
## [1] 300 11
```

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  $\text{CH}_4$  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")
```



### 3.2 Calculating BMP from cumulative production

To calculate BMP we need to subtract the contribution of the inoculum to  $\text{CH}_4$  production for each reactor, 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, norm.name = NULL,
##     inoc.m.name = "minoc", vol.name = "cvCH4",
##     imethod = "linear", extrap = FALSE, when = 30,
##     show.obs = 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<sub>4</sub> 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    sd n
## 1      A    30 1610.7 38.236 3
## 2    inoc    30 1577.6 33.308 3
## 3      B    30 2081.5 46.013 3
## 4    cellu    30 3693.0 57.811 3
```

Here, the response variable was `cvCH4` (cumulative CH<sub>4</sub> 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 reactor. Here it is used for grouping reactors. 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 reactor 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<sub>4</sub> 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<sub>4</sub> production at 60 days (`when` argument).

```
BMP <- summBg(cum.prod, setup = setup, time.name = "days",
             inoc.name = "inoc", inoc.m.name = "minoc",
             norm.name = "mvs.sub", when = 60)

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

BMP
##   descrip days   mean    sd n
## 1      A    60 166.46  6.6278 3
## 2      B    60 142.19 10.9890 3
## 3    cellu    60 408.88 15.3610 3
```

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 reactors, and check the apparent contribution of the inoculum to each reactor's biogas production. This additional information can be returned by setting `show.obs = TRUE`.

### 3.3 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 ( $C_6H_{10}O_5$ ), we can calculate theoretical methane potential to compare to our measurements<sup>4</sup>.

```
predBg("C6H10O5")
## [1] 413.735
```

So we see that theoretical methane potential of cellulose is  $414 \text{ mL g}^{-1}$ . 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    sd n
## 1      A    60 166.46  6.6278 3
## 2      B    60 142.19 10.9890 3
## 3  cellu    60 408.88 15.3610 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] 372.3615
```

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  $1.561 \text{ g g}^{-1}$ . Cellulose has a calculated oxygen demand (COD')<sup>5</sup> of  $1.184 \text{ g g}^{-1}$ . Predicted  $\text{CH}_4$  production per g VS is therefore:

```
predBg(COD = c(A = 1.439, B = 1.561, cellu = 1.184))
##      A      B    cellu
## 502.7731 545.3988 413.6785
```

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

```
BMP$mean/predBg(COD = c(A = 1.439, B = 1.561, cellu = 1.184))
##      A      B    cellu
## 0.3310837 0.2607083 0.9884004
```

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

<sup>4</sup> 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  $\text{CH}_4$ , as described in [3].

<sup>5</sup> Oxygen demand can be calculated with the `calcCOD` function.



once before these measurements, and so we should expect low degradability. We can conclude that are measured results are reasonable.

## 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 reactor 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. The low-level functions are straight-forward to use, and details can also be found in the help files.

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] Sasha D. Hafner, Charlotte Rennuit, Jin Mi Triolo, and Brian K. Richards. Validation of a simple gravimetric method for measuring biogas production in laboratory experiments. *Biomass and Bioenergy*, In review.
- [2] BK Richards, RJ Cummings, TE White, and WJ Jewell. Methods for kinetic-analysis of methane fermentation in high solids biomass digesters. *Biomass & Bioenergy*, 1(2):65–73, 1991.
- [3] Bruce E. Rittmann and Perry L. McCarty. *Environmental Biotechnology: Principles and Applications*. McGraw-Hill series in water resources and environmental engineering. McGraw-Hill, Boston, 2001.