Online Analytical Chemistry notes: data analysis part II

Sasha D. Hafner

10:49 - 08 March, 2024

Contents

Overview

These notes are on analysis of an intervention or perturbation in time series data. You might generate data like these when evaluating the temporary effect of some treatment on the concentration or production of a particular compound.

Main ideas

1. Autocorrelation makes statistical analysis more difficult
2. Lurking variables can cause problems
3. We can make inferences without statistical models (at least about one experimental unit)
4. Time series methods exist but are not simple and can (should?) be avoided
5. Repetition at experimental unit level is (still) important for inferences
6. Response feature analysis works well for assessing interventions

Autocorrelation

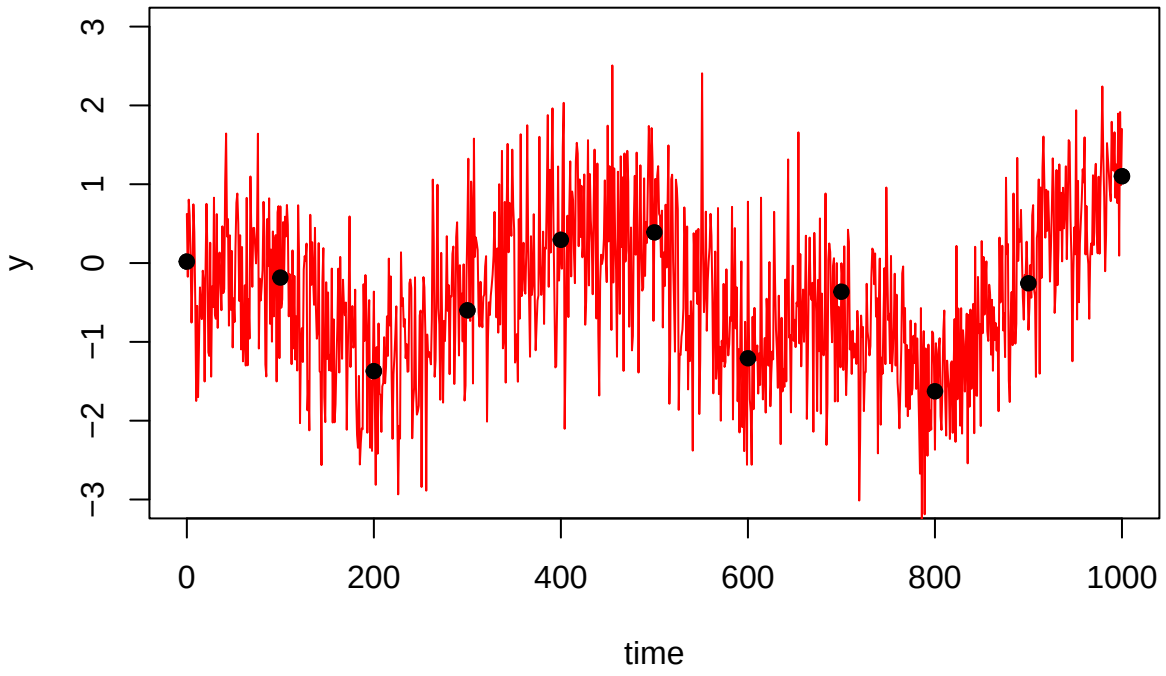
Autocorrelation is just correlation between adjacent measurements. It is commonly present in time series type data. Autocorrelation could have multiple sources, e.g., instrument drift, reaction or mass transfer causing changes over time, mixing in a reactor (retention time effect), change in ambient temperature or other conditions, ...

Here are some generated autocorrelated data.

```
library(data.table)
```

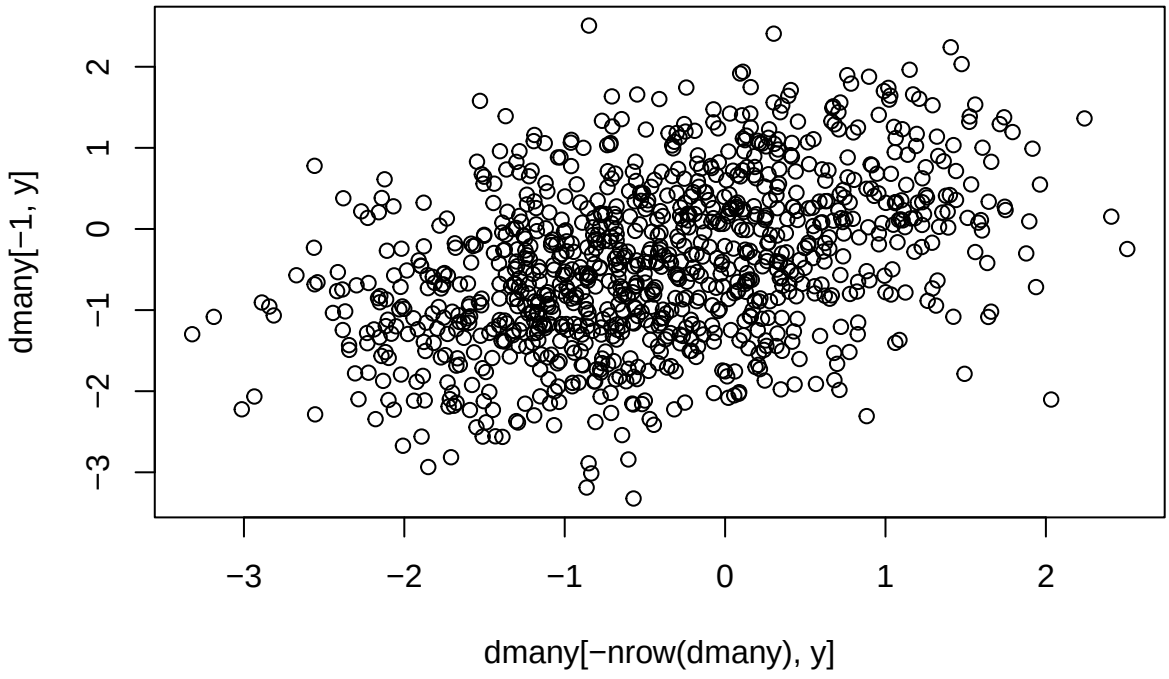
```
## data.table 1.14.8 using 2 threads (see ?getDTthreads).  Latest news: r-datatable.com
```

```r
set.seed(10)
dfew <- data.table(time = 0:10 * 100, y = rnorm(11))
plot(y ~ time, data = dfew, pch = 19, ylim = c(-3, 3))
dmany <- data.table(time = 0:1000, y = approx(dfew[, time], dfew[, y], xout = 0:1000)$y + rnorm(1001, s
lines(y ~ time, data = dmany, col = 'red')
points(y ~ time, data = dfew, pch = 19)
```



See the correlation? Find it by

```r
plot(dmany[-nrow(dmany), y], dmany[-1, y])
```

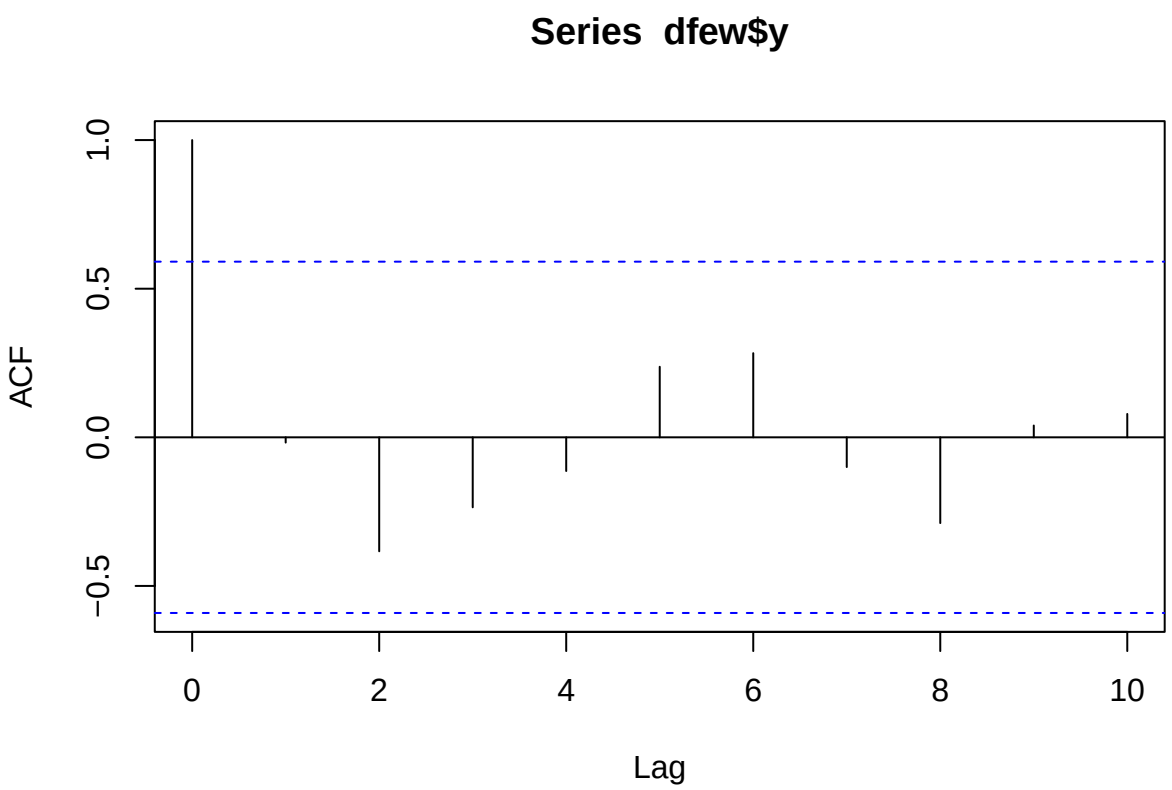


```
cor(dmany[-nrow(dmany), y], dmany[-1, y])
```

```
## [1] 0.4059568
```

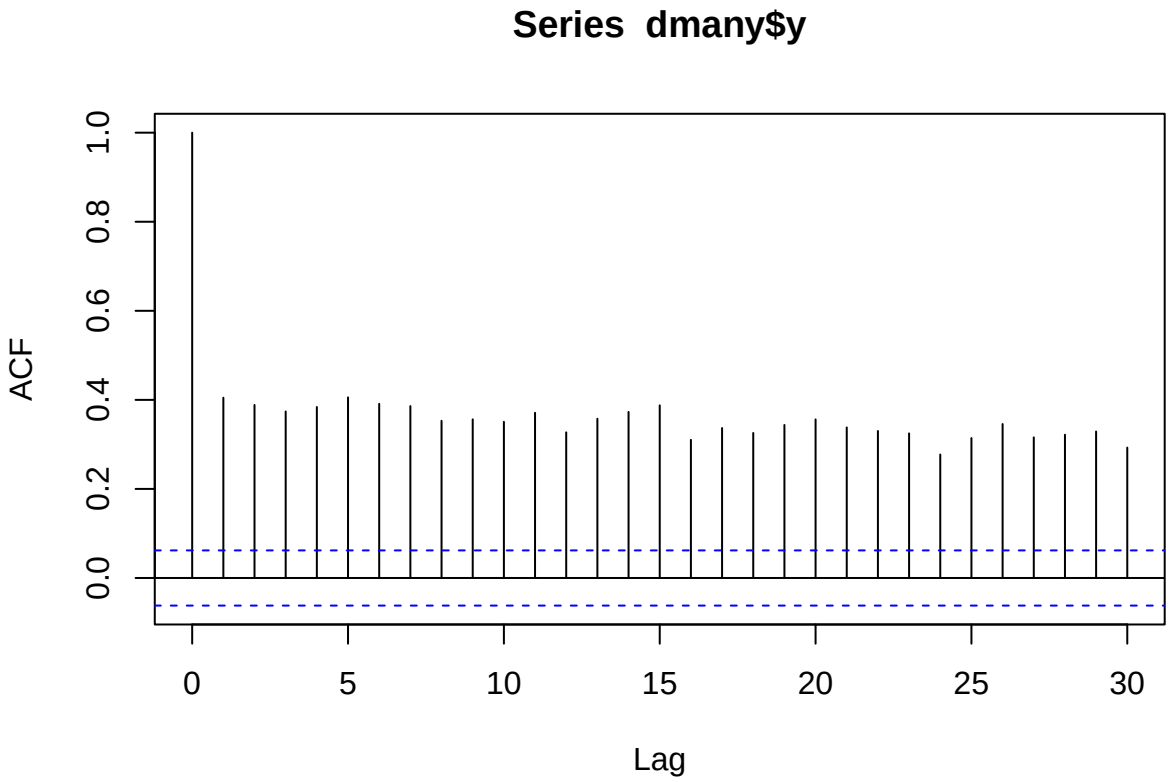
In R the `acf()` function can be used to check. It returns the correlation between the data and lagged versions.

```
(acf(dfew$y))
```




```
##
## Autocorrelations of series 'dfew$y', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000 -0.018 -0.383 -0.236 -0.113  0.237  0.283 -0.100 -0.289  0.040  0.079
```

```
(acf(dmany$y))
```




```
##
## Autocorrelations of series 'dmany$y', by lag
##
##     0     1     2     3     4     5     6     7     8     9    10    11    12
## 1.000 0.405 0.389 0.374 0.384 0.406 0.391 0.386 0.353 0.356 0.351 0.371 0.327
##    13    14    15    16    17    18    19    20    21    22    23    24    25
## 0.358 0.373 0.388 0.310 0.337 0.326 0.344 0.356 0.338 0.330 0.325 0.277 0.314
##    26    27    28    29    30
## 0.346 0.316 0.322 0.329 0.293
```

The presence of autocorrelation violates the assumption of independent observations. So it is important that measurements are not treated as indepdendent.

Lurking variables

Lurking variables are some variables that affect the response but are not measured or considered in an analysis. They are probably more of a problem for observational data than experimental data. But a combination of a lurking variable and autocorrelation could make for very wrong inferences! Here is an example.

```
library(data.table)
```

```
amm_int <- fread('../data/NH3_emis_acid_interval.csv')
amm_int
```

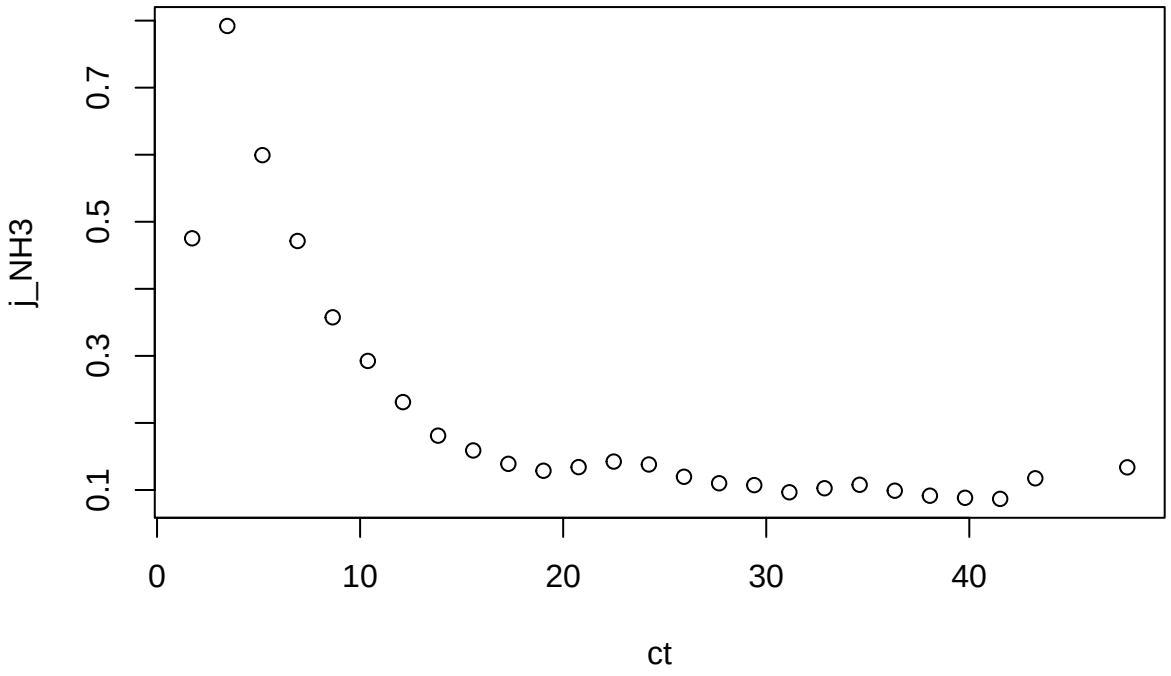
```
##       pmid     ct      cta   dt             t_start               t_end
##    1: 1947   1.73   1.7333 1.73 2020-11-18 13:40:00 2020-11-18 15:24:00
##    2: 1947   3.46   3.4667 1.73 2020-11-18 15:24:00 2020-11-18 17:08:00
##    3: 1947   5.19   5.2000 1.73 2020-11-18 17:08:00 2020-11-18 18:52:00
##    4: 1947   6.92   6.9333 1.73 2020-11-18 18:52:00 2020-11-18 20:36:00
##    5: 1947   8.65   8.6667 1.73 2020-11-18 20:36:00 2020-11-18 22:20:00
```

```
##    ---
## 3485: 1982 178.19 178.5300 1.73 2020-12-16 23:49:00 2020-12-17 01:33:00
## 3486: 1982 179.92 180.2700 1.73 2020-12-17 01:33:00 2020-12-17 03:17:00
## 3487: 1982 181.65 182.0000 1.73 2020-12-17 03:17:00 2020-12-17 05:01:00
## 3488: 1982 183.38 183.7300 1.73 2020-12-17 05:01:00 2020-12-17 06:45:00
## 3489: 1982 185.11 185.4700 1.73 2020-12-17 06:45:00 2020-12-17 08:29:00
##           j_NH3 air_temp
##    1: 0.0088216   11.470
##    2: 0.0000000   11.600
##    3: 0.0061700   10.510
##    4: 0.0136090   10.240
##    5: 0.0154260   10.560
##    ---
## 3485: 0.0100490    5.818
## 3486: 0.0098460    5.763
## 3487: 0.0095709    5.917
## 3488: 0.0099536    6.172
## 3489: 0.0116350    6.791
```

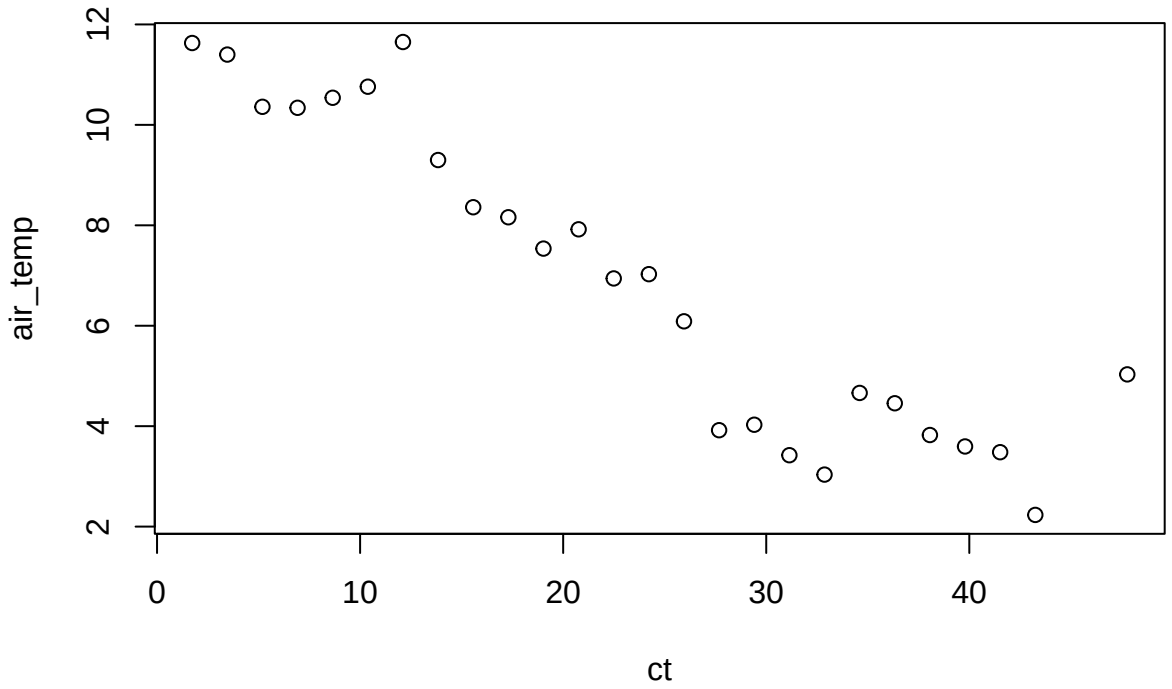
```
amm_sub <- amm_int[ct < 48 & pmid == 1951, ]
```

Emission decreased over the first couple days and so did temperature. Did the change in temperature cause the drop in emission?

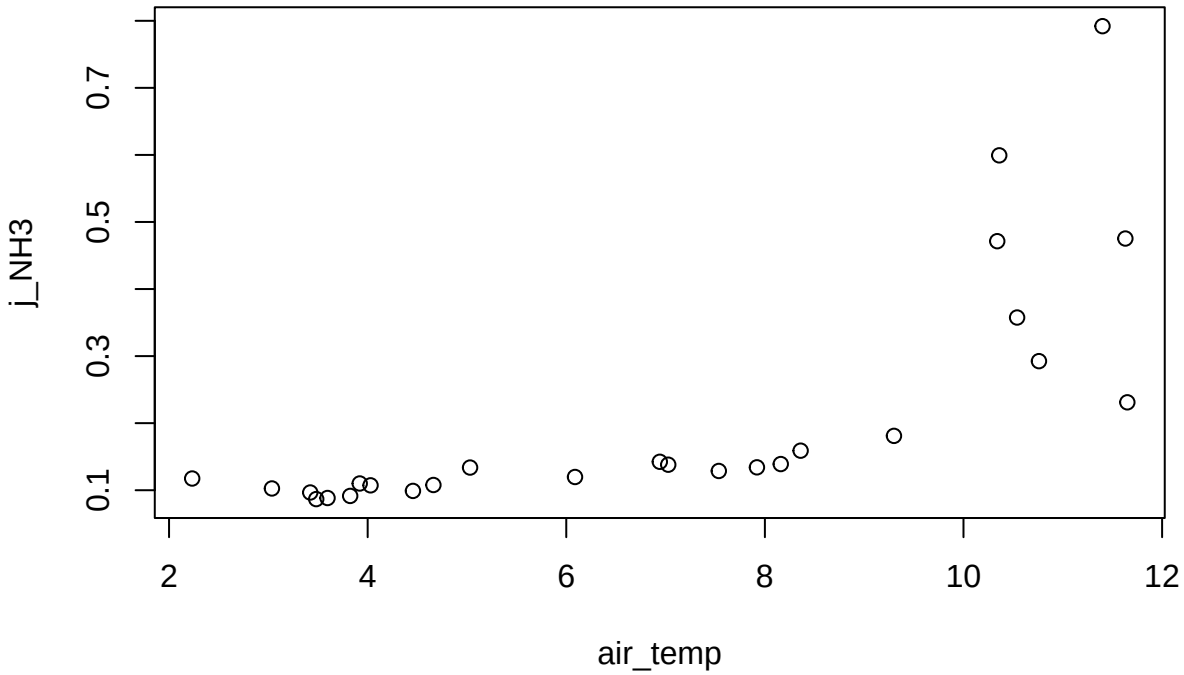
```
plot(j_NH3 ~ ct, data = amm_sub)
```




```
plot(air_temp ~ ct, data = amm_sub)
```




```
plot(j_NH3 ~ air_temp, data = amm_sub)
```



Let's fit a (stupid!) model.

```
mod1 <- lm(j_NH3 ~ air_temp, data = amm_sub)
summary(mod1)
```

```
##
## Call:
## lm(formula = j_NH3 ~ air_temp, data = amm_sub)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.18560 -0.08390 -0.00142  0.03396  0.38613
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.087489   0.060474  -1.447    0.161
## air_temp     0.043275   0.008007   5.405  1.5e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1243 on 24 degrees of freedom
## Multiple R-squared:  0.549,  Adjusted R-squared:  0.5302
## F-statistic: 29.21 on 1 and 24 DF,  p-value: 1.496e-05
```

```
mod2 <- lm(log10(j_NH3) ~ air_temp, data = amm_sub)
summary(mod2)
```

```
##
## Call:
## lm(formula = log10(j_NH3) ~ air_temp, data = amm_sub)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22435 -0.08768 -0.01825  0.05138  0.32997
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.31326    0.07009 -18.736 7.84e-16 ***
## air_temp     0.07737    0.00928   8.337 1.51e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1441 on 24 degrees of freedom
## Multiple R-squared:  0.7433, Adjusted R-squared:  0.7326
## F-statistic: 69.51 on 1 and 24 DF,  p-value: 1.508e-08
```

That's a very low p value. This is bad bad bad practice!

Inferences without statistics

Here are some data from a recent paper.

IOP Publishing Environ. Res. Lett. 19 (2024) 014017 https://doi.org/10.1088/1748-9326/ad0e33

ENVIRONMENTAL RESEARCH
LETTERS

OPEN ACCESS

RECEIVED
11 September 2023

REVISED
30 October 2023

ACCEPTED FOR PUBLICATION
20 November 2023

PUBLISHED
18 December 2023

LETTER

A high efficiency gas phase photoreactor for eradication of methane from low-concentration sources

Morten Krogsbøll[1], Hugo S Russell[1,2] and Matthew S Johnson[1,3,*]

[1] Ambient Carbon ApS, Forhaabningsholms Alle 19, 1.th, DK-1904 Frederiksberg C, Denmark
[2] Department of Environmental Science, Aarhus University, Frederiksborgvej 399, DK-4000 Roskilde, Denmark
[3] Department of Chemistry, University of Copenhagen, Universitetsparken 5, DK-2100 Copenhagen OE, Denmark
[*] Author to whom any correspondence should be addressed.

E-mail: msj@chem.ku.dk

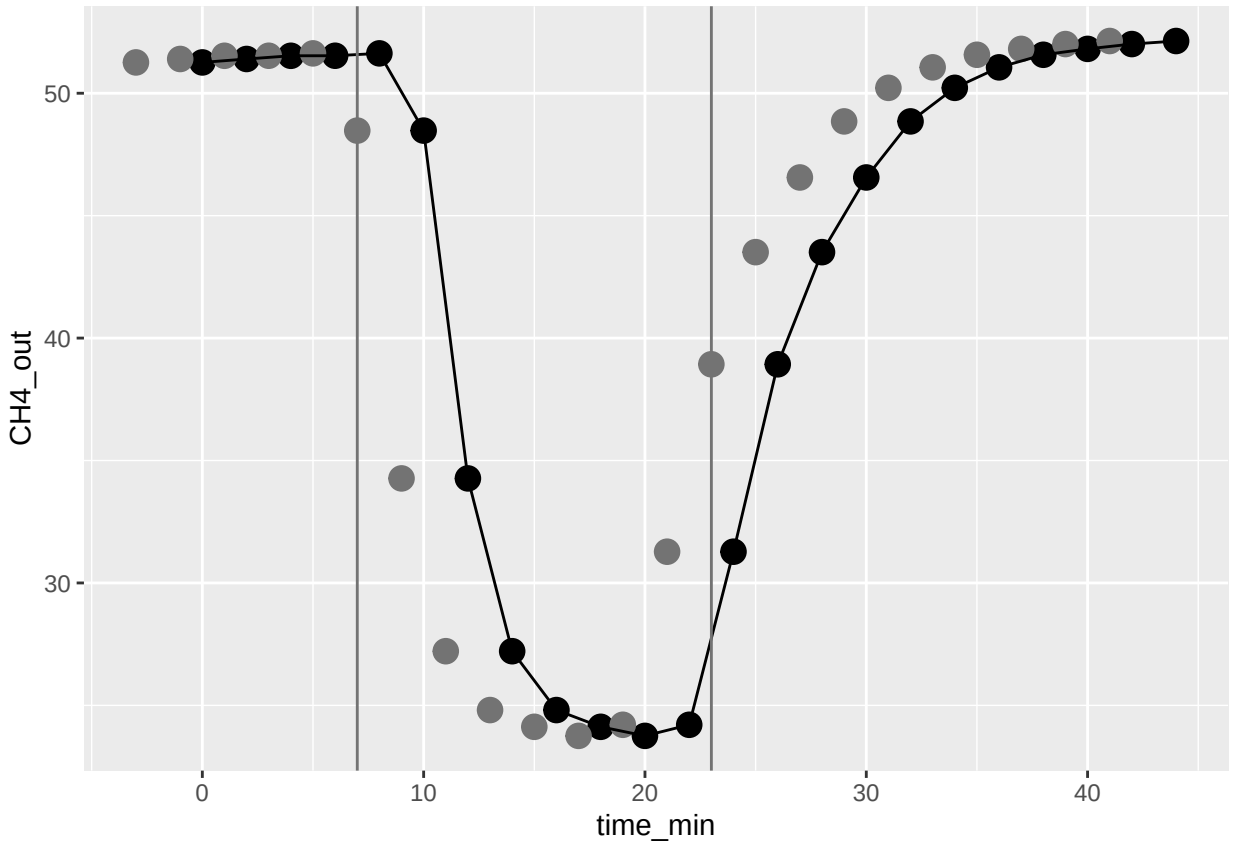
```
library(data.table)
```

```
pr <- fread('../data/photo_reactor.csv')
pr[, led := factor(led)]
head(pr)
```

```
##              date_time  C12_out    C12_in  CH4_out   CH4_in time_min led
## 1: 2023-06-09 12:15:00 102.2350 102.06809 51.25882 51.30022        0   0
## 2: 2023-06-09 12:17:00 101.4895 101.29243 51.39887 51.78323        2   0
## 3: 2023-06-09 12:19:00 101.1526  99.82636 51.52996 51.88457        4   0
## 4: 2023-06-09 12:21:00 100.6248 100.54421 51.53150 51.87321        6   0
## 5: 2023-06-09 12:23:00 100.0695 100.10595 51.62850 51.97208        8   1
## 6: 2023-06-09 12:25:00  89.7600 100.15390 48.47417 52.33748       10   1
```

```
library(ggplot2)
```

```
ggplot(pr, aes(time_min, CH4_out)) +
  geom_point(size = 4) +
  geom_point(aes(x = I(time_min - 3)), colour = 'gray45', size = 4) +
  geom_line() +
  geom_vline(xintercept = c(7, 23), colour = 'gray45')
```



The purpose of the photoreactor is to destroy methane. LED lights were on from around 7 to 23 minutes (see vertical lines). The authors concluded–without application of a statistical model–that these measurements show that the light drive about destruction of about 50% of the methane coming into the reactor. Note that they also presented results from other runs, typically shown as average removal efficiency, i.e., a response feature instead of measured dynamics.

Time series methods

There are some statistical methods designed for analyzing time series data. Most seem to have a focus on prediction and perhaps less so on hypothesis testing. ARMA and ARIMA are approaches that include a moving average component. There are also approaches for assessing interventions or perturbations. If you are interested, the book by McDowall et al. (2019) is not a bad place to start, but it is not focused on implementation in R or Python. R also has a "task view" (collection and summary of available packages) here: https://cran.r-project.org/web/views/TimeSeries.html. Most of the methods are probably not relevant. The term for the type of analysis we would be interested in is "interrupted time series analysis", ITSA in McDowall et al. There is an R package focused on just this called its.analysis: https://cran.r-project.org/web/packages/its.analysis/index.html. But it uses a new and simple method that might have some limitations. For Python there are functions in the statsmodels module that can be used. Here is a detailed tutorial on ITSA: https://www.xboard.dev/posts/2020_01_01_interrupted-time-series-python-part-I/.

But, I do not recommend these approaches. First, they are not particularly simple to understand or implement. More importantly, they are focused on analysis of a single experimental unit. In classical time series work, this makes sense–there was only one state of California from 1980-2000. But for our type of work, it does not make as much sense. Inferences that apply to a single experimental unit are not so useful. Better to repeat, as described in the next section.

Repeat experiments

As we discussed last week, statistical inference with any really useul scope really requires repeated application of a treatment or intervention to individual experimental units, i.e., multiple experimental units. So if you can, repeat. Then response feature analysis can easily be used. Here we will create a larger dataset from the one photoreactor run.

```
pr <- fread('../data/photo_reactor.csv')
pr[, led := factor(led)]
head(pr)
```

```
##              date_time  Cl2_out    Cl2_in  CH4_out   CH4_in time_min led
## 1: 2023-06-09 12:15:00 102.2350 102.06809 51.25882 51.30022        0   0
## 2: 2023-06-09 12:17:00 101.4895 101.29243 51.39887 51.78323        2   0
## 3: 2023-06-09 12:19:00 101.1526  99.82636 51.52996 51.88457        4   0
## 4: 2023-06-09 12:21:00 100.6248 100.54421 51.53150 51.87321        6   0
## 5: 2023-06-09 12:23:00 100.0695 100.10595 51.62850 51.97208        8   1
## 6: 2023-06-09 12:25:00  89.7600 100.15390 48.47417 52.33748       10   1
```

We'll make a fake dataset here with 5 separate runs, all operated the same way.

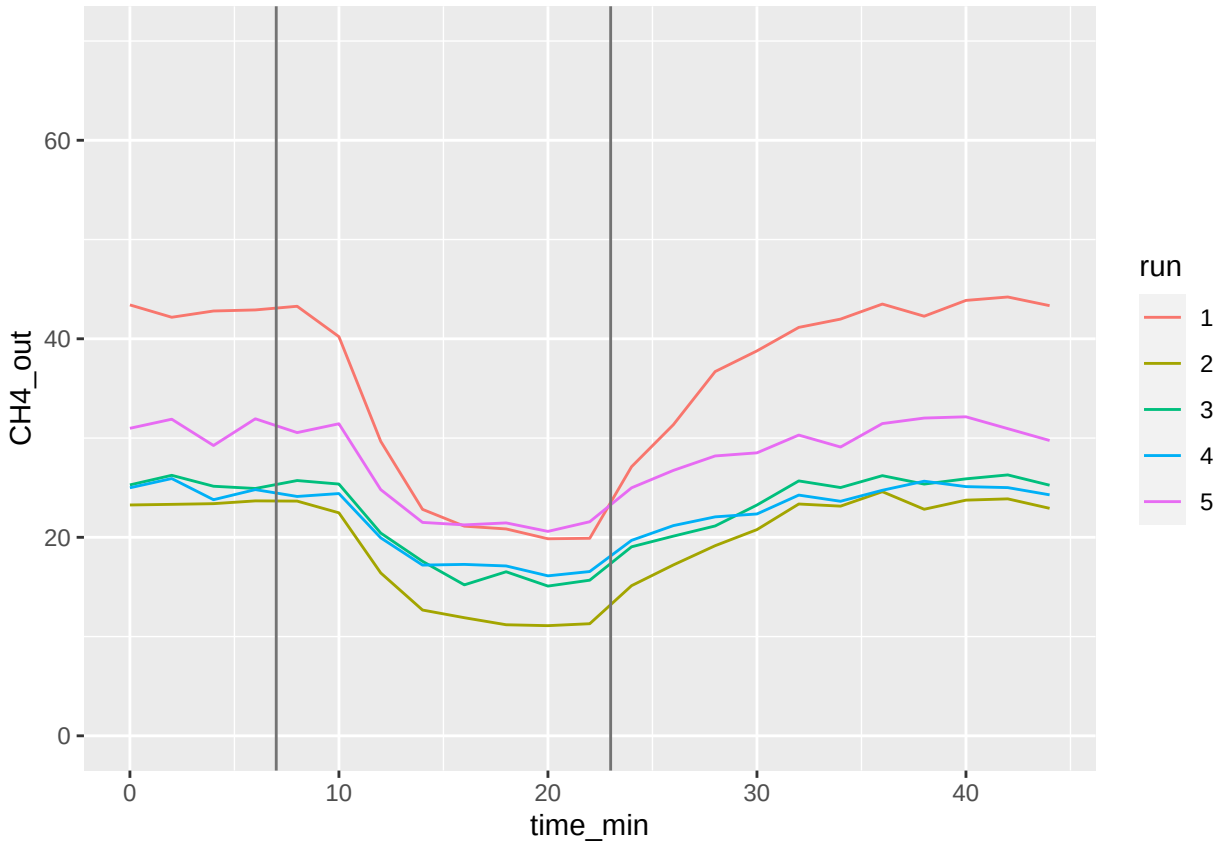
```
prm <- data.table()

n <- 5

set.seed(2)
for (i in 1:n) {
  d <- pr
  d[, run := i]
  d[, CH4_out := CH4_out * rnorm(1, mean = 1, sd = 0.2) + rnorm(1, sd = 3) + rnorm(nrow(pr), sd = 0.5)]
  prm <- rbind(prm, d)
}

prm[, run := factor(run)]
```

```
ggplot(prm, aes(time_min, CH4_out, colour = run)) +
  geom_line() +
  geom_vline(xintercept = c(7, 23), colour = 'gray45') +
  ylim(0, 70)
```



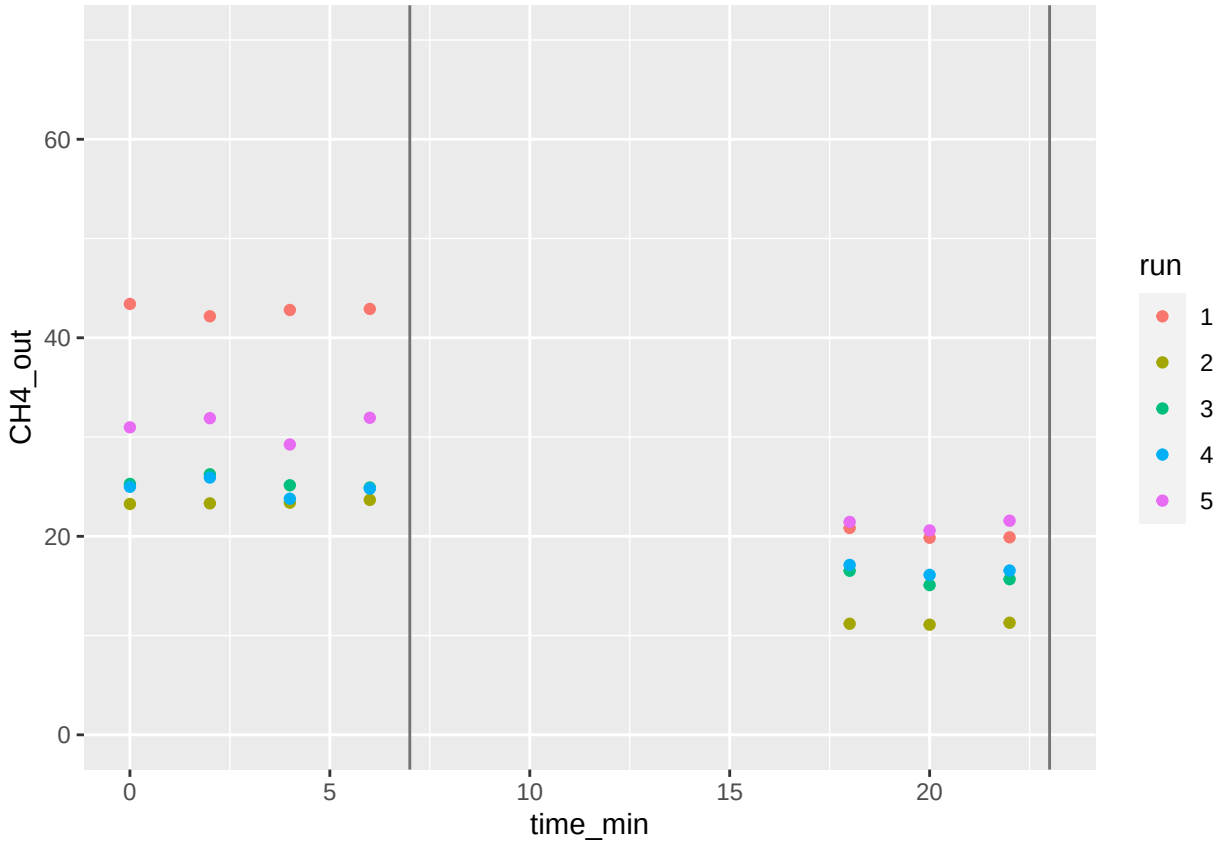

```
table(prm[, .(time_min, led)])
```

```
##          led
## time_min 0 1
##        0  5 0
##        2  5 0
##        4  5 0
##        6  5 0
##        8  0 5
##        10 0 5
##        12 0 5
##        14 0 5
##        16 0 5
##        18 0 5
##        20 0 5
##        22 0 5
##        24 5 0
##        26 5 0
##        28 5 0
##        30 5 0
##        32 5 0
##        34 5 0
##        36 5 0
##        38 5 0
##        40 5 0
##        42 5 0
##        44 5 0
```

Let's try to cut out the decay and build up of methane, i.e., target what seem to be steady-state concentrations (although really we should probably have longer runs for this).

```
pra <- prm[time_min < 8 | (time_min > 16 & time_min < 24), ]
```

```
ggplot(pra, aes(time_min, CH4_out, colour = run)) +
  geom_point() +
  geom_vline(xintercept = c(7, 23), colour = 'gray45') +
  ylim(0, 70)
```



Now mean by run.

```
prm <- pra[, .(CH4_out_mn = mean(CH4_out)), by = .(run, led)]
prm
```

```
##     run led CH4_out_mn
##  1:   1   0   42.82241
##  2:   1   1   20.19788
##  3:   2   0   23.40956
##  4:   2   1   11.19447
##  5:   3   0   25.39618
##  6:   3   1   15.76424
##  7:   4   0   24.87557
##  8:   4   1   16.59396
##  9:   5   0   31.01816
## 10:   5   1   21.20504
```

And fit model, using run as blocking variable.

```r
mod1 <- lm(CH4_out_mn ~ led + run, data = prm)
summary(mod1)
```

```
##
## Call:
## lm(formula = CH4_out_mn ~ led + run, data = prm)
##
## Residuals:
##       1       2       3       4       5       6       7       8       9      10
##  5.0556 -5.0556 -0.1491  0.1491 -1.4407  1.4407 -2.1158  2.1158 -1.3501  1.3501
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   37.767      3.192  11.833 0.000292 ***
## led1         -12.513      2.606  -4.802 0.008638 **
## run2         -14.208      4.120  -3.448 0.026098 *
## run3         -10.930      4.120  -2.653 0.056828 .
## run4         -10.775      4.120  -2.615 0.059104 .
## run5          -5.399      4.120  -1.310 0.260308
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.12 on 4 degrees of freedom
## Multiple R-squared:  0.9043, Adjusted R-squared:  0.7847
## F-statistic: 7.562 on 5 and 4 DF,  p-value: 0.03628
```

```r
anova(mod1)
```

```
## Analysis of Variance Table
##
## Response: CH4_out_mn
##           Df Sum Sq Mean Sq F value   Pr(>F)
## led        1 391.45  391.45 23.0561 0.008638 **
## run        4 250.50   62.63  3.6886 0.117066
## Residuals  4  67.91   16.98
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The method used to extract a feature from an individual run (experimental unit) could be just about anything. It might be fitting a nonlinear model and taking one of the coefficients, for example.