

Online Analytical Chemistry notes: data analysis

Sasha D. Hafner

10:38 29 February, 2024

Overview

These notes are on analysis of data, including application of statistical models, that comes after data manipulation.

Multiple measurements on one experimental unit

We'll start with an important problem that runs through all this material: hypothesis testing or statistical inference requires replication at the experimental unit level. The *experimental unit* is the physical thing to which an experimental treatment is applied. Without this kind of replication, even millions of measurements will tell us nothing about an effect or a difference *in general*. Making multiple measurements on a single experimental unit results in data that may be called *structured* (an unfortunately vague term, perhaps). You might also see the terms *hierarchical*, *nested*, or *subsampling* used. In this case, the lack of *independence* among observations becomes a problem. Unfortunately, as Bello and Renter wrote:

"Yet, a staggering number of standard statistical methods commonly used for data analysis, particularly those taught in introductory statistics courses (e.g., correlations, z-tests, t-tests, traditional ANOVA, regression), implicitly assume that observations are mutually independent and that data structure is nonexistent. Most concerning, these critical assumptions often go understated, thus effectively disregarding any correlation patterns in the data induced by the hierarchical structure of data architecture. As a direct consequence, experimental replication is naively matched one-to-one with level of observation in the data (Stroup, 2013) and the inferential space is misrepresented (Tempelman, 2009). This oversight can have serious downstream implications for inference (Aitkin and Longford, 1986) and provide a false sense of security on results, meanwhile undermining RR."

This all relates to the *scope of inference*, or the population to which a statistical inference applies. While we might generally like our results to apply to a very broad population, a narrow one is more likely. And without replication of experimental units, the scope becomes very narrow.

Let's try to keep these general issues in mind as we explore some ways to analyze data generated from online measurements. But also, be sure to keep them in mind for design of experiments. Try to think about the ideal scope of inference and how replication at the experimental unit level can be done.

General ideas

For data analysis we will focus on relatively simple methods that can handle structured data. Some of the main ideas are:

1. Statistical models are not always needed
2. Statistical models can be difficult to appropriately apply *directly* to measurement data
3. Recognition of the experimental unit is essential
4. Response feature analysis is simple and circumvents problems with structured data
5. Nonlinear regression can be helpful for fitting complex models or feature extraction

6. Mixed-effects models are an easy and flexible approach for working with structured data

When it comes to statistical modeling, we will go over these four groups of methods:

1. Response feature analysis
2. Classical linear models
3. Mixed-effects models
4. Nonlinear regression

These four are related; some are used together.

No statistics needed

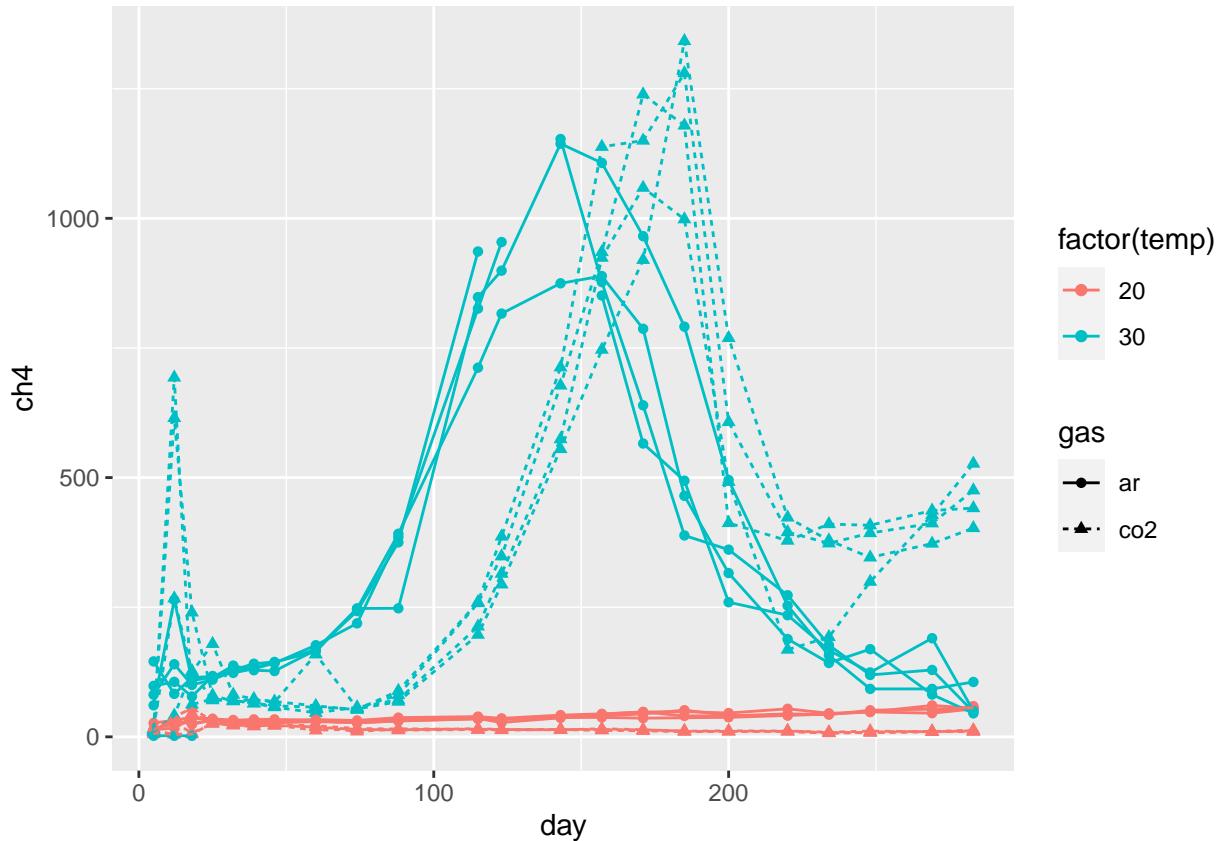
1. Methane emission in lab incubations

```
library(data.table)
dat <- fread('../data/slurry_emis.csv')
dat

##      reactor     ch4      co2      flow day gas temp
## 1:      R1 11.374 338.300 0.06300000 5 co2 20
## 2:      R2  9.638 348.235 0.07300000 5 co2 20
## 3:      R3  5.221 320.180 0.08200000 5 co2 20
## 4:      R4  7.200 313.690 0.08100000 5 co2 20
## 5:      R5 16.000 371.500 0.08400000 5 co2 30
## ---
## 350:    R12 59.150 1002.000 0.06121372 283 ar 20
## 351:    R13 48.320  858.300 0.06754617 283 ar 30
## 352:    R14 49.970  865.400 0.06860158 283 ar 30
## 353:    R15 45.260  837.200 0.06860158 283 ar 30
## 354:    R16 105.800  895.000 0.05910290 283 ar 30

library(ggplot2)
ggplot(dat, aes(day, ch4, colour = factor(temp), shape = gas, group = reactor, lty = gas)) +
  geom_point() + geom_line()

## Warning: Removed 6 rows containing missing values (`geom_point()`).
```



2. Ammonia volatilization dynamics

```

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

##      pmid     ct     cta    dt          t_start          t_end      j_NH3
## 1: 1947 1.73 1.7333 1.73 2020-11-18 13:40:00 2020-11-18 15:24:00 0.0088216
## 2: 1947 3.46 3.4667 1.73 2020-11-18 15:24:00 2020-11-18 17:08:00 0.0000000
## 3: 1947 5.19 5.2000 1.73 2020-11-18 17:08:00 2020-11-18 18:52:00 0.0061700
## 4: 1947 6.92 6.9333 1.73 2020-11-18 18:52:00 2020-11-18 20:36:00 0.0136090
## 5: 1947 8.65 8.6667 1.73 2020-11-18 20:36:00 2020-11-18 22:20:00 0.0154260
##   --
## 3485: 1982 178.19 178.5300 1.73 2020-12-16 23:49:00 2020-12-17 01:33:00 0.0100490
## 3486: 1982 179.92 180.2700 1.73 2020-12-17 01:33:00 2020-12-17 03:17:00 0.0098460
## 3487: 1982 181.65 182.0000 1.73 2020-12-17 03:17:00 2020-12-17 05:01:00 0.0095709
## 3488: 1982 183.38 183.7300 1.73 2020-12-17 05:01:00 2020-12-17 06:45:00 0.0099536
## 3489: 1982 185.11 185.4700 1.73 2020-12-17 06:45:00 2020-12-17 08:29:00 0.0116350

amm_plot <- fread('~/data/NH3_emis_acid_plot.csv')
amm_plot

##      pmid treat app_date tan_app e_cum_final e_rel_final date_int
## 1: 1947 tank 2020-11-18    97.30     3.9108    0.040193      1
## 2: 1948 tank 2020-11-18    97.30     4.9536    0.050910      1
## 3: 1949 field 2020-11-18   103.60    13.6860    0.132110      1
## 4: 1950 field 2020-11-18   103.60    12.3270    0.118980      1
## 5: 1951 none 2020-11-18    95.20    20.0020    0.210100      1

```

```

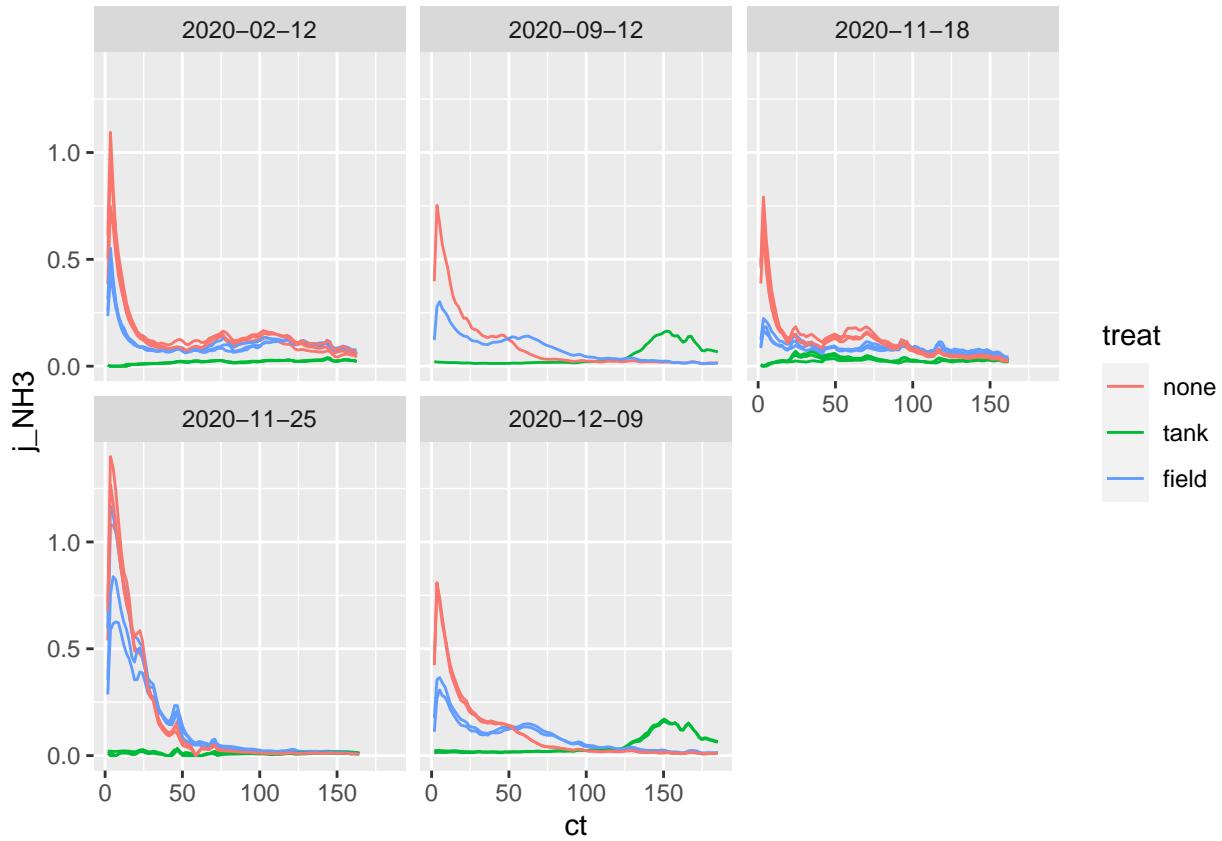
##   6: 1952 field 2020-11-18 103.60    14.6960  0.141860    1
##   7: 1953 none 2020-11-18  95.20    19.9610  0.209670    1
##   8: 1954 tank 2020-11-18  97.30     5.3328  0.054808    1
##   9: 1955 none 2020-11-18  95.20    17.1320  0.179960    1
##  10: 1956 none 2020-11-25  71.75    25.1850  0.351020    2
##  11: 1957 field 2020-11-25  72.45    26.9790  0.372390    2
##  12: 1958 tank 2020-11-25  67.55     1.3104  0.019399    2
##  13: 1959 field 2020-11-25  72.45    20.7570  0.286510    2
##  14: 1960 tank 2020-11-25  67.55     1.8739  0.027741    2
##  15: 1961 none 2020-11-25  71.75    25.3840  0.353780    2
##  16: 1962 tank 2020-11-25  67.55     2.3160  0.034286    2
##  17: 1963 field 2020-11-25  72.45    23.5660  0.325270    2
##  18: 1964 none 2020-11-25  71.75    26.8990  0.374900    2
##  19: 1965 none 2020-02-12 151.20    20.4720  0.135400    3
##  20: 1966 tank 2020-02-12 118.30     3.3581  0.028386    3
##  21: 1967 field 2020-02-12 149.10    17.5260  0.117540    3
##  22: 1968 field 2020-02-12 149.10    17.5560  0.117750    3
##  23: 1969 tank 2020-02-12 118.30     3.1914  0.026977    3
##  24: 1970 field 2020-02-12 149.10    17.2320  0.115580    3
##  25: 1971 none 2020-02-12 151.20    25.9790  0.171820    3
##  26: 1972 tank 2020-02-12 118.30     3.1087  0.026278    3
##  27: 1973 none 2020-02-12 151.20    24.6010  0.162700    3
##  28: 1974 tank 2020-09-12  71.40     8.6166  0.120680    3
##  29: 1975 tank 2020-12-09  71.40     8.8196  0.123520    4
##  30: 1976 field 2020-12-09  65.10    15.6990  0.241150    4
##  31: 1977 none 2020-09-12  66.50    17.2490  0.259380    4
##  32: 1978 field 2020-09-12  65.10    14.6140  0.224490    4
##  33: 1979 none 2020-12-09  66.50    18.9850  0.285480    4
##  34: 1980 tank 2020-12-09  71.40     9.3760  0.131320    4
##  35: 1981 field 2020-12-09  65.10    14.6650  0.225270    4
##  36: 1982 none 2020-12-09  66.50    18.4340  0.277210    4
##      pmid treat app_date tan_app e_cum_final e_rel_final date_int
amm_plot[, treat := factor(treat, levels = c('none', 'tank', 'field'))]

amm <- merge(amm_plot, amm_int, by = 'pmid')
amm

##      pmid treat app_date tan_app e_cum_final e_rel_final date_int      ct      cta      dt
##  1: 1947 tank 2020-11-18    97.3    3.9108  0.040193    1  1.73  1.7333 1.73 2020-11-18
##  2: 1947 tank 2020-11-18    97.3    3.9108  0.040193    1  3.46  3.4667 1.73 2020-11-18
##  3: 1947 tank 2020-11-18    97.3    3.9108  0.040193    1  5.19  5.2000 1.73 2020-11-18
##  4: 1947 tank 2020-11-18    97.3    3.9108  0.040193    1  6.92  6.9333 1.73 2020-11-18
##  5: 1947 tank 2020-11-18    97.3    3.9108  0.040193    1  8.65  8.6667 1.73 2020-11-18
##  --
## 3485: 1982 none 2020-12-09    66.5    18.4340  0.277210    4 178.19 178.5300 1.73 2020-12-16
## 3486: 1982 none 2020-12-09    66.5    18.4340  0.277210    4 179.92 180.2700 1.73 2020-12-17
## 3487: 1982 none 2020-12-09    66.5    18.4340  0.277210    4 181.65 182.0000 1.73 2020-12-17
## 3488: 1982 none 2020-12-09    66.5    18.4340  0.277210    4 183.38 183.7300 1.73 2020-12-17
## 3489: 1982 none 2020-12-09    66.5    18.4340  0.277210    4 185.11 185.4700 1.73 2020-12-17

library(ggplot2)
ggplot(amm, aes(ct, j_NH3, group = factor(pmid), colour = treat)) +
  geom_line() +
  facet_wrap(~ app_date)

```



Response feature analysis

The basic idea with response feature analysis is to extract or calculate some characteristic or *feature* from the multiple measurements made on an individual experimental unit. This process collapses what might be a large number of measurements to a single value—typically one per experimental unit. These values then become the observations you use in a statistical analysis. For datasets that show simple responses over time, it could be the slope and intercept that are used. The more complex patterns typical in our online measurements might suggest responses like the integrated totals, peak values, time of maxima, or even the simple mean value. Nonlinear regression could be used to extract some rate constant or other model parameter. Ideally the selected response should be one that is relevant and interesting—not just something that facilitates hypothesis testing. Selecting a response after viewing results runs the risk of invalidating any p values or other statistical results. So be careful to avoid too much “fishing”. Here is an example with the data we have seen before where our response is integrated total emission.

Integration for response feature analysis

```
library(data.table)

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

##      pmid     ct     cta    dt      t_start      t_end     j_NH3
## 1: 1947 1.73 1.7333 1.73 2020-11-18 13:40:00 2020-11-18 15:24:00 0.0088216
## 2: 1947 3.46 3.4667 1.73 2020-11-18 15:24:00 2020-11-18 17:08:00 0.0000000
## 3: 1947 5.19 5.2000 1.73 2020-11-18 17:08:00 2020-11-18 18:52:00 0.0061700
## 4: 1947 6.92 6.9333 1.73 2020-11-18 18:52:00 2020-11-18 20:36:00 0.0136090
```

```

##      5: 1947    8.65    8.6667 1.73 2020-11-18 20:36:00 2020-11-18 22:20:00 0.0154260
##     ---
## 3485: 1982 178.19 178.5300 1.73 2020-12-16 23:49:00 2020-12-17 01:33:00 0.0100490
## 3486: 1982 179.92 180.2700 1.73 2020-12-17 01:33:00 2020-12-17 03:17:00 0.0098460
## 3487: 1982 181.65 182.0000 1.73 2020-12-17 03:17:00 2020-12-17 05:01:00 0.0095709
## 3488: 1982 183.38 183.7300 1.73 2020-12-17 05:01:00 2020-12-17 06:45:00 0.0099536
## 3489: 1982 185.11 185.4700 1.73 2020-12-17 06:45:00 2020-12-17 08:29:00 0.0116350

amm_plot <- fread('../data/NH3_emis_acid_plot.csv')
amm_plot

##      pmid treat   app_date tan_app e_cum_final e_rel_final date_int
## 1: 1947 tank 2020-11-18    97.30     3.9108    0.040193      1
## 2: 1948 tank 2020-11-18    97.30     4.9536    0.050910      1
## 3: 1949 field 2020-11-18   103.60    13.6860    0.132110      1
## 4: 1950 field 2020-11-18   103.60    12.3270    0.118980      1
## 5: 1951 none 2020-11-18    95.20    20.0020    0.210100      1
## 6: 1952 field 2020-11-18   103.60    14.6960    0.141860      1
## 7: 1953 none 2020-11-18    95.20    19.9610    0.209670      1
## 8: 1954 tank 2020-11-18    97.30     5.3328    0.054808      1
## 9: 1955 none 2020-11-18    95.20    17.1320    0.179960      1
## 10: 1956 none 2020-11-25    71.75    25.1850    0.351020      2
## 11: 1957 field 2020-11-25    72.45    26.9790    0.372390      2
## 12: 1958 tank 2020-11-25    67.55     1.3104    0.019399      2
## 13: 1959 field 2020-11-25    72.45    20.7570    0.286510      2
## 14: 1960 tank 2020-11-25    67.55     1.8739    0.027741      2
## 15: 1961 none 2020-11-25    71.75    25.3840    0.353780      2
## 16: 1962 tank 2020-11-25    67.55     2.3160    0.034286      2
## 17: 1963 field 2020-11-25    72.45    23.5660    0.325270      2
## 18: 1964 none 2020-11-25    71.75    26.8990    0.374900      2
## 19: 1965 none 2020-02-12   151.20    20.4720    0.135400      3
## 20: 1966 tank 2020-02-12   118.30     3.3581    0.028386      3
## 21: 1967 field 2020-02-12   149.10    17.5260    0.117540      3
## 22: 1968 field 2020-02-12   149.10    17.5560    0.117750      3
## 23: 1969 tank 2020-02-12   118.30     3.1914    0.026977      3
## 24: 1970 field 2020-02-12   149.10    17.2320    0.115580      3
## 25: 1971 none 2020-02-12   151.20    25.9790    0.171820      3
## 26: 1972 tank 2020-02-12   118.30     3.1087    0.026278      3
## 27: 1973 none 2020-02-12   151.20    24.6010    0.162700      3
## 28: 1974 tank 2020-09-12    71.40     8.6166    0.120680      3
## 29: 1975 tank 2020-12-09    71.40     8.8196    0.123520      4
## 30: 1976 field 2020-12-09    65.10    15.6990    0.241150      4
## 31: 1977 none 2020-09-12    66.50    17.2490    0.259380      4
## 32: 1978 field 2020-09-12    65.10    14.6140    0.224490      4
## 33: 1979 none 2020-12-09    66.50    18.9850    0.285480      4
## 34: 1980 tank 2020-12-09    71.40     9.3760    0.131320      4
## 35: 1981 field 2020-12-09    65.10    14.6650    0.225270      4
## 36: 1982 none 2020-12-09    66.50    18.4340    0.277210      4

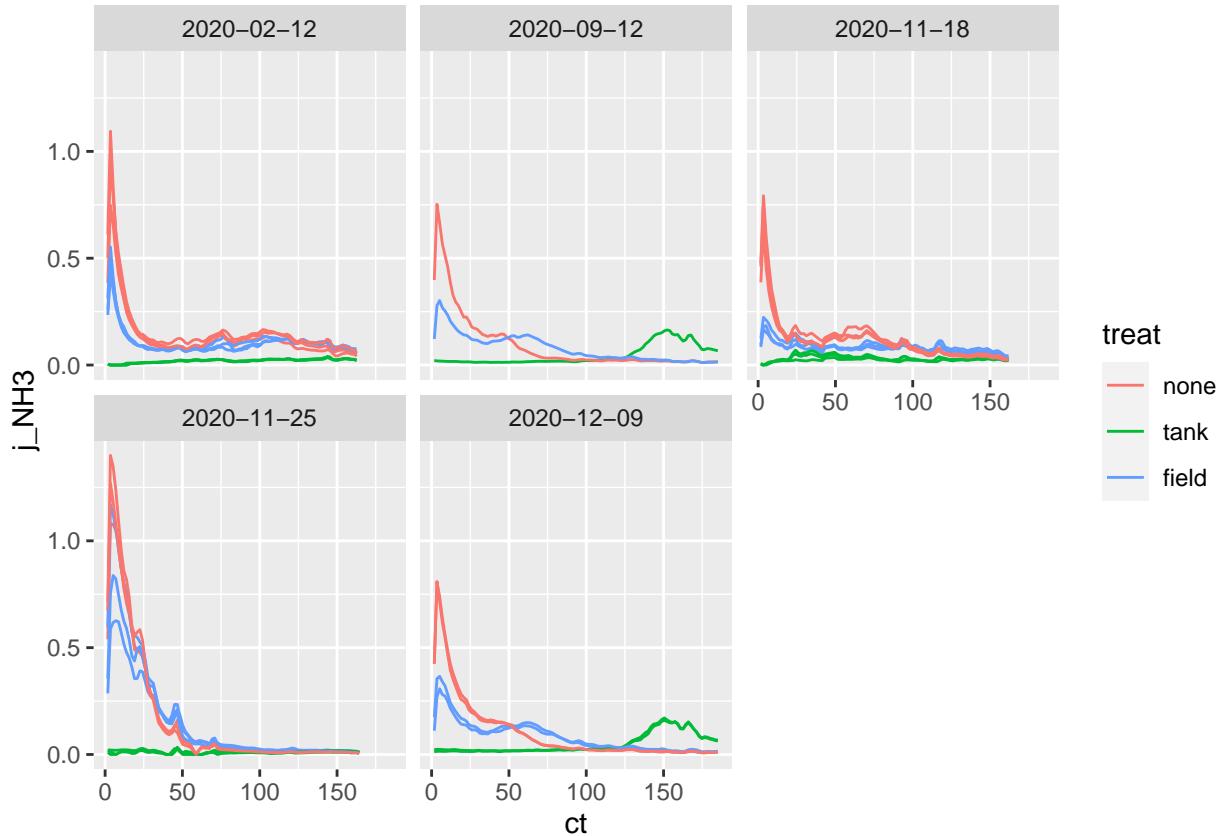
##      pmid treat   app_date tan_app e_cum_final e_rel_final date_int
amm_plot[, treat := factor(treat, levels = c('none', 'tank', 'field'))]
amm_plot[, app_date := factor(app_date)]

```

```
amm <- merge(amm_plot, amm_int, by = 'pmid')
amm
```

```
##      pmid treat app_date tan_app e_cum_final e_rel_final date_int     ct      cta      dt
## 1: 1947 tank 2020-11-18   97.3    3.9108  0.040193      1 1.73 1.7333 1.73 2020-11-18
## 2: 1947 tank 2020-11-18   97.3    3.9108  0.040193      1 3.46 3.4667 1.73 2020-11-18
## 3: 1947 tank 2020-11-18   97.3    3.9108  0.040193      1 5.19 5.2000 1.73 2020-11-18
## 4: 1947 tank 2020-11-18   97.3    3.9108  0.040193      1 6.92 6.9333 1.73 2020-11-18
## 5: 1947 tank 2020-11-18   97.3    3.9108  0.040193      1 8.65 8.6667 1.73 2020-11-18
##   ---
## 3485: 1982 none 2020-12-09   66.5   18.4340  0.277210      4 178.19 178.5300 1.73 2020-12-16
## 3486: 1982 none 2020-12-09   66.5   18.4340  0.277210      4 179.92 180.2700 1.73 2020-12-17
## 3487: 1982 none 2020-12-09   66.5   18.4340  0.277210      4 181.65 182.0000 1.73 2020-12-17
## 3488: 1982 none 2020-12-09   66.5   18.4340  0.277210      4 183.38 183.7300 1.73 2020-12-17
## 3489: 1982 none 2020-12-09   66.5   18.4340  0.277210      4 185.11 185.4700 1.73 2020-12-17

library(ggplot2)
ggplot(amm, aes(ct, j_NH3, group = factor(pmids), colour = treat)) +
  geom_line() +
  facet_wrap(~ app_date)
```



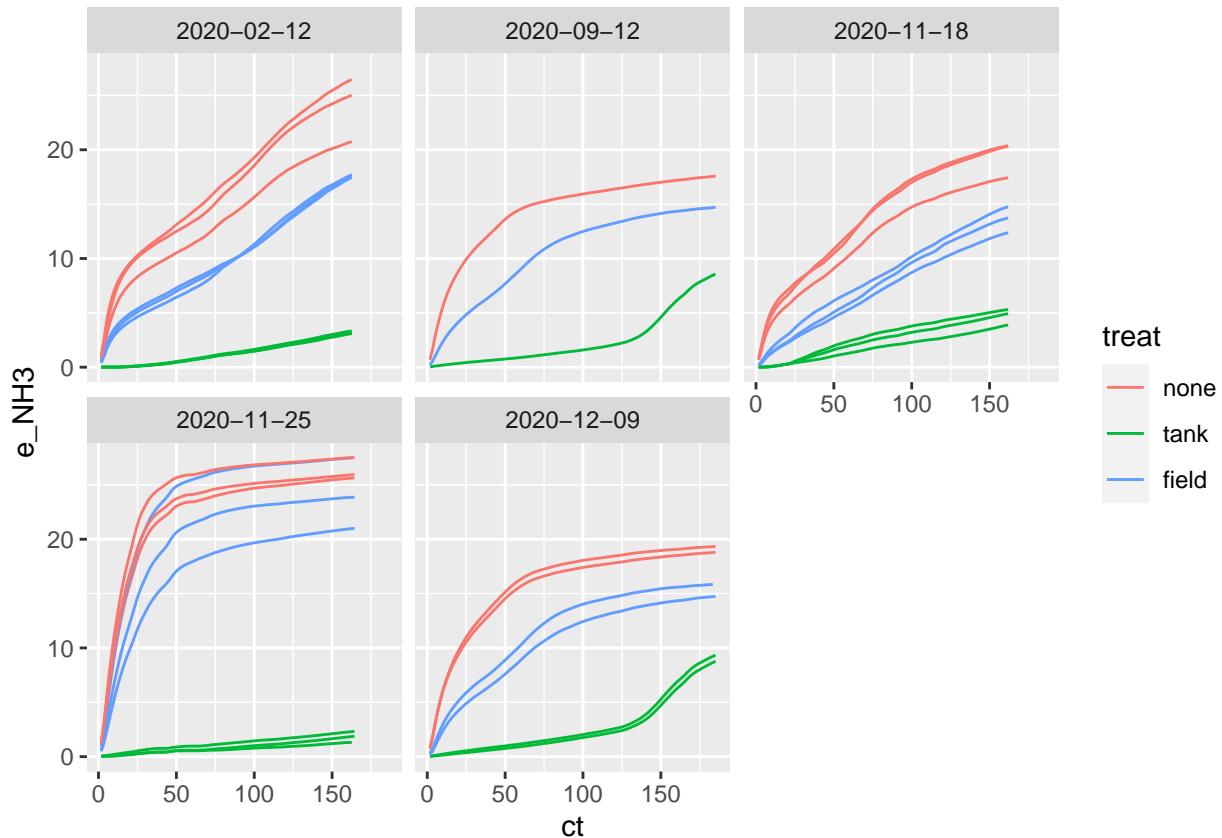
```
source('../R-functions/mintegrate.R')
args(mintegrate)

## function (x, y, method = "midpoint", lwr = min(x), upr = max(x),
##         ylwr = y[which.min(x)], value = "all")
## NULL
```

```
amm[, e_NH3 := mintegrate(ct, j_NH3, method = 'trap', lwr = 0), by = pmid]
amm
```

```
##      pmid treat app_date tan_app e_cum_final e_rel_final date_int     ct      cta      dt
## 1: 1947 tank 2020-11-18    97.3     3.9108  0.040193      1 1.73 1.7333 1.73 2020-11-18
## 2: 1947 tank 2020-11-18    97.3     3.9108  0.040193      1 3.46 3.4667 1.73 2020-11-18
## 3: 1947 tank 2020-11-18    97.3     3.9108  0.040193      1 5.19 5.2000 1.73 2020-11-18
## 4: 1947 tank 2020-11-18    97.3     3.9108  0.040193      1 6.92 6.9333 1.73 2020-11-18
## 5: 1947 tank 2020-11-18    97.3     3.9108  0.040193      1 8.65 8.6667 1.73 2020-11-18
##   ---
## 3485: 1982 none 2020-12-09    66.5    18.4340  0.277210      4 178.19 178.5300 1.73 2020-12-16
## 3486: 1982 none 2020-12-09    66.5    18.4340  0.277210      4 179.92 180.2700 1.73 2020-12-17
## 3487: 1982 none 2020-12-09    66.5    18.4340  0.277210      4 181.65 182.0000 1.73 2020-12-17
## 3488: 1982 none 2020-12-09    66.5    18.4340  0.277210      4 183.38 183.7300 1.73 2020-12-17
## 3489: 1982 none 2020-12-09    66.5    18.4340  0.277210      4 185.11 185.4700 1.73 2020-12-17

ggplot(amm, aes(ct, e_NH3, group = factor(pmrid), colour = treat)) +
  geom_line() +
  facet_wrap(~ app_date)
```



```
table(amm_plot[, .(app_date, treat)])
```

```
##          treat
## app_date   none tank field
## 2020-02-12   3   3   3
## 2020-09-12   1   1   1
## 2020-11-18   3   3   3
## 2020-11-25   3   3   3
```

```

##    2020-12-09      2      2      2
amm_tot <- amm[, .(e_NH3 = mintegrate(ct, j_NH3, method = 'trap', lwr = 0, value = 'total')), by = pmid]
amm_tot

##      pmid      e_NH3
## 1: 1947  3.888339
## 2: 1948  4.924019
## 3: 1949 13.726046
## 4: 1950 12.367994
## 5: 1951 20.369072
## 6: 1952 14.755975
## 7: 1953 20.311890
## 8: 1954  5.307757
## 9: 1955 17.422558
## 10: 1956 25.641761
## 11: 1957 27.487533
## 12: 1958  1.319052
## 13: 1959 20.998197
## 14: 1960  1.872497
## 15: 1961 25.953998
## 16: 1962  2.325047
## 17: 1963 23.870387
## 18: 1964 27.513122
## 19: 1965 20.759140
## 20: 1966  3.342990
## 21: 1967 17.682434
## 22: 1968 17.714323
## 23: 1969  3.174567
## 24: 1970 17.449584
## 25: 1971 26.459916
## 26: 1972  3.093773
## 27: 1973 24.999419
## 28: 1974  8.577115
## 29: 1975  8.772058
## 30: 1976 15.840716
## 31: 1977 17.579123
## 32: 1978 14.707340
## 33: 1979 19.341930
## 34: 1980  9.341222
## 35: 1981 14.748452
## 36: 1982 18.794922
##      pmid      e_NH3
amm_tot <- merge(amm_tot, amm_plot, by = 'pmid')
amm_tot[, e_NH3_rel := 100 * e_NH3 / tan_app]
amm_tot

##      pmid      e_NH3 treat app_date tan_app e_cum_final e_rel_final date_int e_NH3_rel
## 1: 1947  3.888339 tank 2020-11-18  97.30      3.9108     0.040193      1  3.996237
## 2: 1948  4.924019 tank 2020-11-18  97.30      4.9536     0.050910      1  5.060656
## 3: 1949 13.726046 field 2020-11-18 103.60     13.6860     0.132110      1 13.249079
## 4: 1950 12.367994 field 2020-11-18 103.60     12.3270     0.118980      1 11.938218
## 5: 1951 20.369072 none 2020-11-18  95.20     20.0020     0.210100      1 21.396084
## 6: 1952 14.755975 field 2020-11-18 103.60     14.6960     0.141860      1 14.243219

```

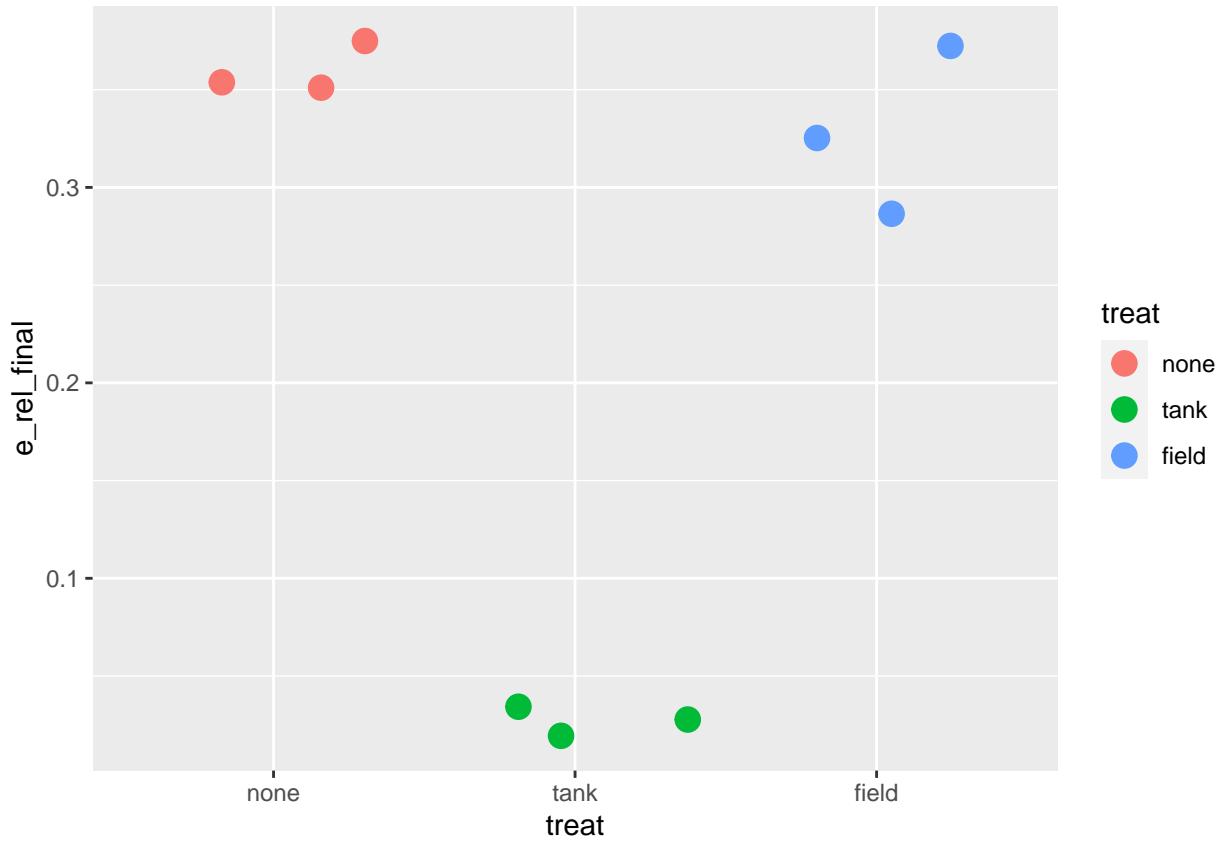
```

## 7: 1953 20.311890 none 2020-11-18 95.20 19.9610 0.209670 1 21.336018
## 8: 1954 5.307757 tank 2020-11-18 97.30 5.3328 0.054808 1 5.455044
## 9: 1955 17.422558 none 2020-11-18 95.20 17.1320 0.179960 1 18.301007
## 10: 1956 25.641761 none 2020-11-25 71.75 25.1850 0.351020 2 35.737646
## 11: 1957 27.487533 field 2020-11-25 72.45 26.9790 0.372390 2 37.940005
## 12: 1958 1.319052 tank 2020-11-25 67.55 1.3104 0.019399 2 1.952704
## 13: 1959 20.998197 field 2020-11-25 72.45 20.7570 0.286510 2 28.983018
## 14: 1960 1.872497 tank 2020-11-25 67.55 1.8739 0.027741 2 2.772016
## 15: 1961 25.953998 none 2020-11-25 71.75 25.3840 0.353780 2 36.172820
## 16: 1962 2.325047 tank 2020-11-25 67.55 2.3160 0.034286 2 3.441964
## 17: 1963 23.870387 field 2020-11-25 72.45 23.5660 0.325270 2 32.947393
## 18: 1964 27.513122 none 2020-11-25 71.75 26.8990 0.374900 2 38.345815
## 19: 1965 20.759140 none 2020-02-12 151.20 20.4720 0.135400 3 13.729590
## 20: 1966 3.342990 tank 2020-02-12 118.30 3.3581 0.028386 3 2.825858
## 21: 1967 17.682434 field 2020-02-12 149.10 17.5260 0.117540 3 11.859446
## 22: 1968 17.714323 field 2020-02-12 149.10 17.5560 0.117750 3 11.880834
## 23: 1969 3.174567 tank 2020-02-12 118.30 3.1914 0.026977 3 2.683489
## 24: 1970 17.449584 field 2020-02-12 149.10 17.2320 0.115580 3 11.703276
## 25: 1971 26.459916 none 2020-02-12 151.20 25.9790 0.171820 3 17.499944
## 26: 1972 3.093773 tank 2020-02-12 118.30 3.1087 0.026278 3 2.615193
## 27: 1973 24.999419 none 2020-02-12 151.20 24.6010 0.162700 3 16.534007
## 28: 1974 8.577115 tank 2020-09-12 71.40 8.6166 0.120680 3 12.012766
## 29: 1975 8.772058 tank 2020-12-09 71.40 8.8196 0.123520 4 12.285796
## 30: 1976 15.840716 field 2020-12-09 65.10 15.6990 0.241150 4 24.332897
## 31: 1977 17.579123 none 2020-09-12 66.50 17.2490 0.259380 4 26.434772
## 32: 1978 14.707340 field 2020-09-12 65.10 14.6140 0.224490 4 22.591920
## 33: 1979 19.341930 none 2020-12-09 66.50 18.9850 0.285480 4 29.085609
## 34: 1980 9.341222 tank 2020-12-09 71.40 9.3760 0.131320 4 13.082943
## 35: 1981 14.748452 field 2020-12-09 65.10 14.6650 0.225270 4 22.655071
## 36: 1982 18.794922 none 2020-12-09 66.50 18.4340 0.277210 4 28.263041
##      pmid     e_NH3 treat    app_date tan_app e_cum_final e_rel_final date_int e_NH3_rel
```

We will use a single field trial, to avoid getting too complicated at the start.

```
amm1 <- amm_tot[app_date == '2020-11-25', ]
```

```
ggplot(amm1, aes(treat, e_rel_final, colour = treat)) +
  geom_jitter(height = 0, size = 4)
```



Classical linear approach

In R, the function `lm()` (for linear model) gives you access to classical linear models.

```
mod0 <- lm(e_rel_final ~ treat, data = amm1)
summary(mod0)

##
## Call:
## lm(formula = e_rel_final ~ treat, data = amm1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.041547 -0.007743 -0.002787  0.007144  0.044333 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 0.35990   0.01519  23.697 3.71e-07 ***
## treattank   -0.33276   0.02148 -15.492 4.58e-06 ***
## treatfield  -0.03184   0.02148  -1.483   0.189    
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.02631 on 6 degrees of freedom
## Multiple R-squared:  0.9799, Adjusted R-squared:  0.9732 
## F-statistic: 146.2 on 2 and 6 DF,  p-value: 8.135e-06
```

The `aov()` function really just calls it up and returns output that is handled conveniently when using factors.

```

mod1 <- aov(e_rel_final ~ treat, data = amm1)
summary(mod1)

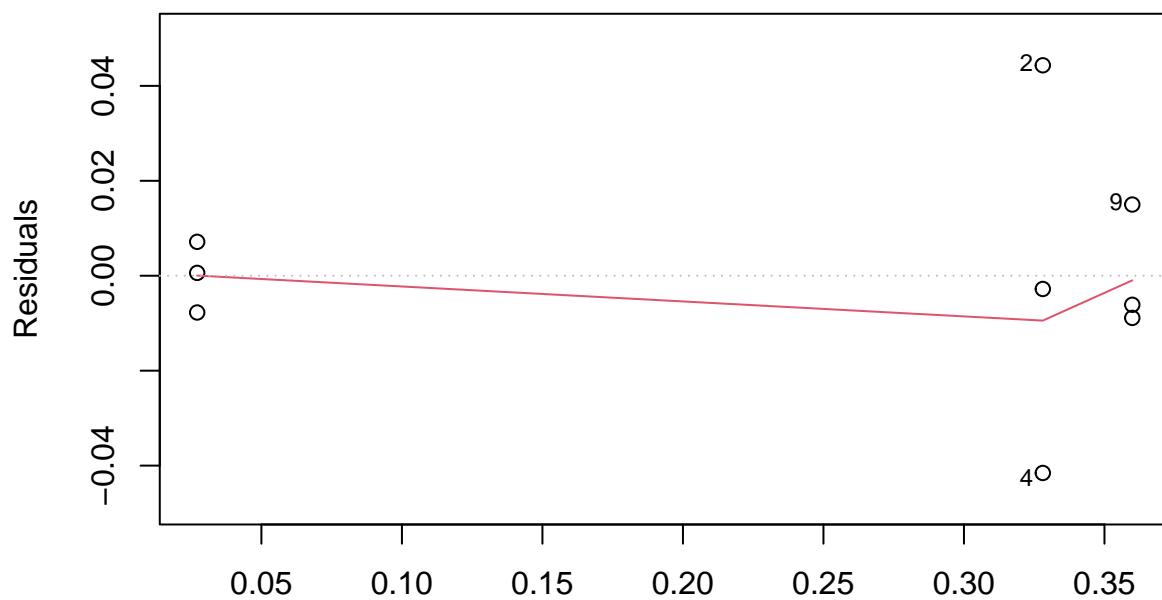
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## treat      2 0.20229 0.10115   146.2 8.14e-06 ***
## Residuals  6 0.00415 0.00069
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
model.tables(mod1, 'means')

## Tables of means
## Grand mean
##
## 0.2383662
##
##  treat
## treat
##   none   tank   field
## 0.3599 0.0271 0.3281
TukeyHSD(mod1)

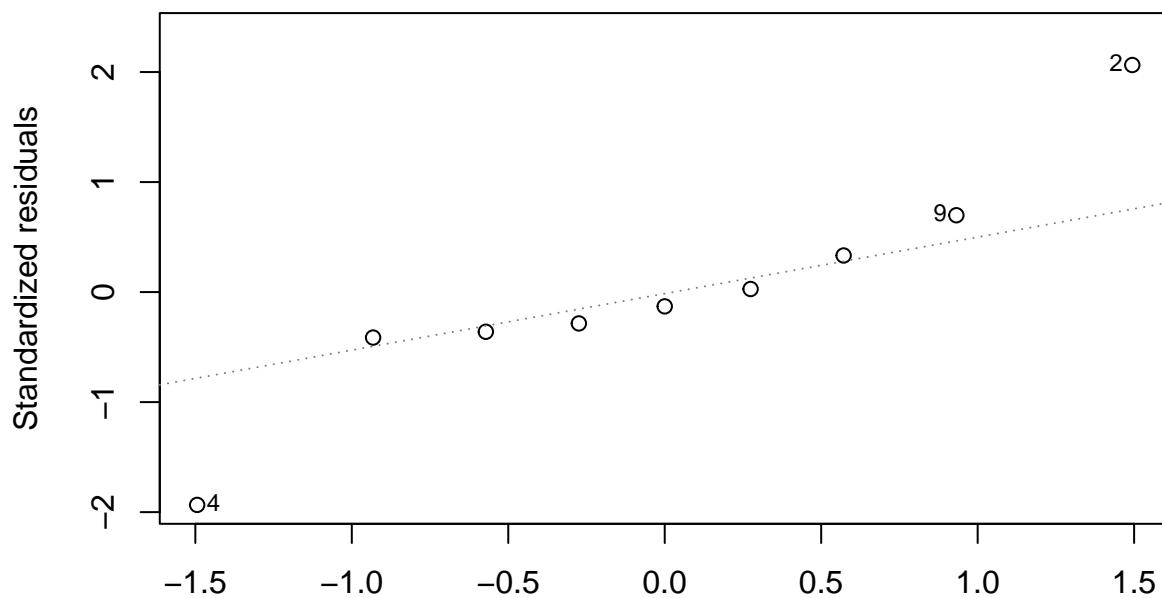
## Tukey multiple comparisons of means
## 95% family-wise confidence level
##
## Fit: aov(formula = e_rel_final ~ treat, data = amm1)
##
## $treat
##        diff      lwr      upr     p adj
## tank-none -0.33275800 -0.39866041 -0.26685559 0.0000109
## field-none -0.03184333 -0.09774574  0.03405908 0.3627117
## field-tank  0.30091467  0.23501226  0.36681708 0.0000201
plot(mod1, ask = FALSE)

```

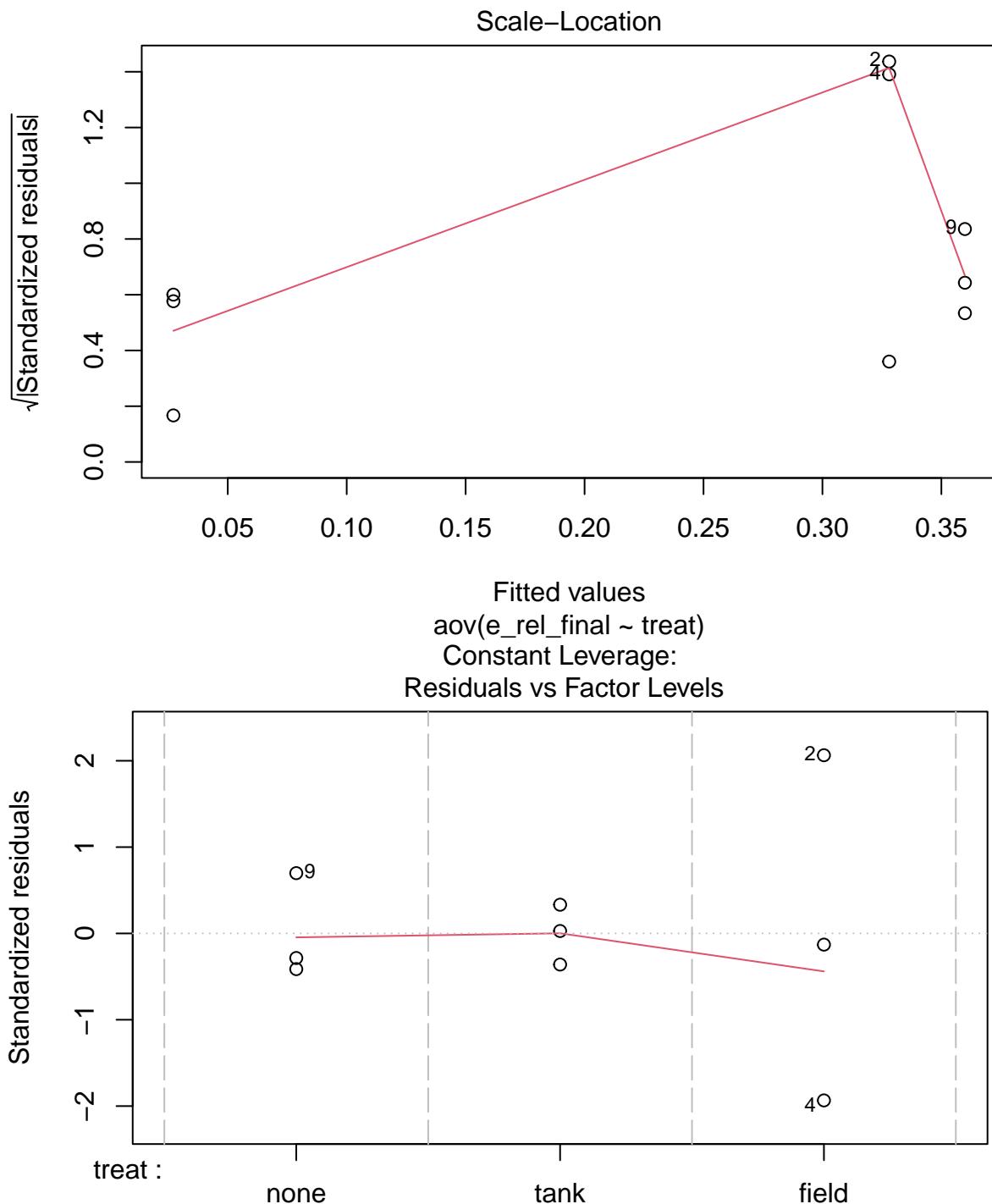
Residuals vs Fitted



Fitted values
aov(e_rel_final ~ treat)
Normal Q-Q



Theoretical Quantiles
aov(e_rel_final ~ treat)



Factor Level Combinations

```
confint(mod1)
```

```
##              2.5 %      97.5 %
## (Intercept) 0.32273703 0.39706297
## treattank   -0.38531438 -0.28020162
## treatfield  -0.08439971  0.02071304
```

In Python, we'll use the statsmodels module.

```
import statsmodels.formula.api as smf

import pandas as pd

amm_plot = pd.read_csv('../data/NH3_emis_acid_plot.csv')
amm_plot['e_rel_final'] = 100 * amm_plot['e_rel_final']
amm1 = amm_plot[amm_plot['app_date'] == '2020-11-25']
print(amm1)
```

```
##      pmid   treat    app_date   tan_app   e_cum_final   e_rel_final   date_int
## 9     1956   none  2020-11-25    71.75     25.1850     35.1020       2
## 10    1957   field 2020-11-25    72.45     26.9790     37.2390       2
## 11    1958   tank  2020-11-25    67.55     1.3104      1.9399       2
## 12    1959   field 2020-11-25    72.45     20.7570     28.6510       2
## 13    1960   tank  2020-11-25    67.55     1.8739      2.7741       2
## 14    1961   none  2020-11-25    71.75     25.3840     35.3780       2
## 15    1962   tank  2020-11-25    67.55     2.3160      3.4286       2
## 16    1963   field 2020-11-25    72.45     23.5660     32.5270       2
## 17    1964   none  2020-11-25    71.75     26.8990     37.4900       2
```

And we'll just use the response variable already given in the plot-level file.

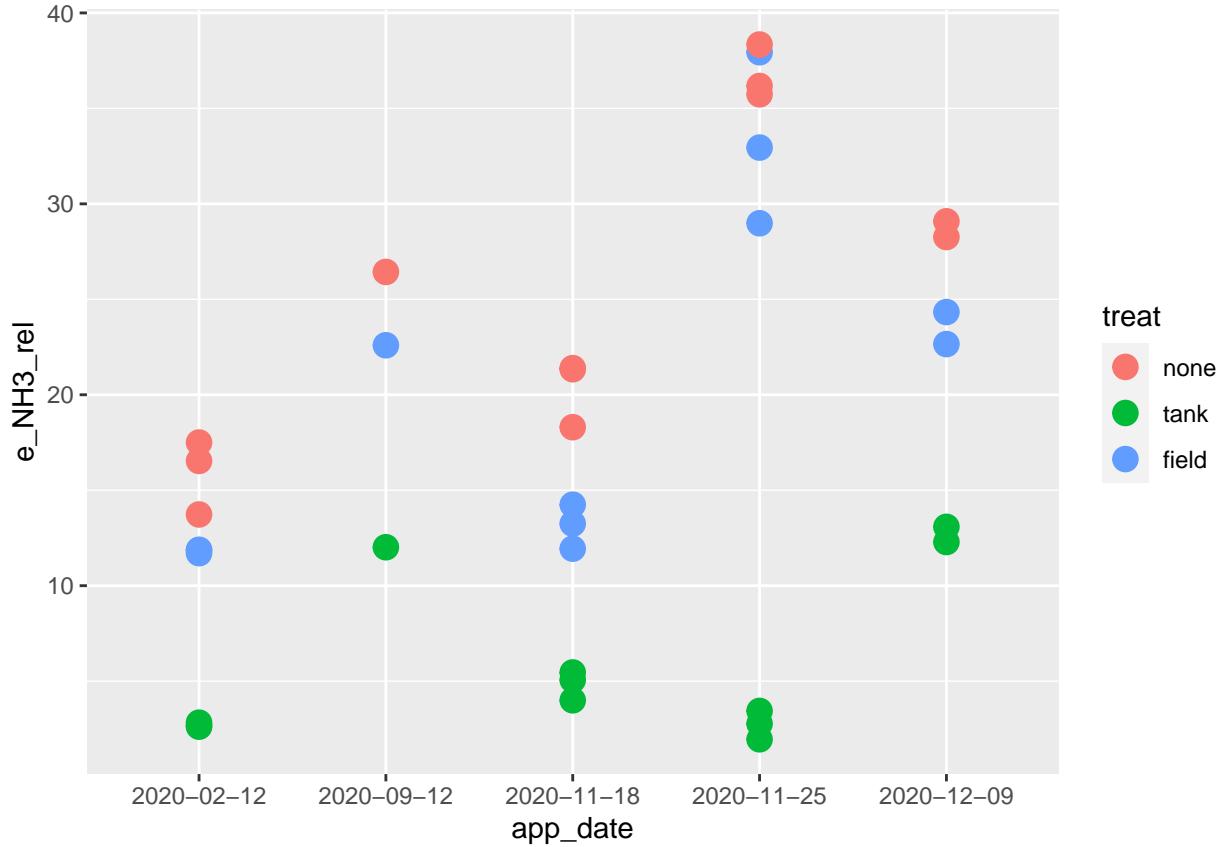
```
mod1 = smf.ols('e_rel_final ~ C(treat, Treatment(reference = "none"))', amm1).fit()
print(mod1.summary())
```

```
##                                     OLS Regression Results
## =====
## Dep. Variable:          e_rel_final   R-squared:                 0.980
## Model:                  OLS           Adj. R-squared:            0.973
## Method:                Least Squares   F-statistic:              146.2
## Date:                  Thu, 29 Feb 2024 Prob (F-statistic):        8.14e-06
## Time:                  10:38:19      Log-Likelihood:           -19.651
## No. Observations:      9             AIC:                      45.30
## Df Residuals:          6             BIC:                      45.89
## Df Model:               2
## Covariance Type:      nonrobust
## =====
##                                         coef    std err      t    P>|t| [0.0]
## -----
## Intercept                           35.9900    1.519   23.697   0.000   32.2
## C(treat, Treatment(reference="none"))[T.field] -3.1843    2.148   -1.483   0.189   -8.4
## C(treat, Treatment(reference="none"))[T.tank]   -33.2758   2.148   -15.492   0.000  -38.5
## =====
## Omnibus:                   2.408   Durbin-Watson:            2.191
## Prob(Omnibus):              0.300   Jarque-Bera (JB):         0.197
## Skew:                       0.199   Prob(JB):                  0.906
## Kurtosis:                   3.606   Cond. No.                  3.73
## =====
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## 
## /home/sasha/.local/share/r-miniconda/envs/r-reticulate/lib/python3.6/site-packages/scipy/stats/stats
## "anyway, n=%i" % int(n))
```

Multiple dates

Can we handle this additional structure?

```
ggplot(amm_tot, aes(app_date, e_NH3_rel, colour = treat)) +
  geom_point(size = 4)
```



amm_tot

```
##      pmid     e_NH3 treat    app_date tan_app e_cum_final e_rel_final date_int e_NH3_rel
## 1: 1947 3.888339 tank 2020-11-18   97.30     3.9108  0.040193       1 3.996237
## 2: 1948 4.924019 tank 2020-11-18   97.30     4.9536  0.050910       1 5.060656
## 3: 1949 13.726046 field 2020-11-18  103.60    13.6860  0.132110       1 13.249079
## 4: 1950 12.367994 field 2020-11-18  103.60    12.3270  0.118980       1 11.938218
## 5: 1951 20.369072 none 2020-11-18   95.20    20.0020  0.210100       1 21.396084
## 6: 1952 14.755975 field 2020-11-18  103.60    14.6960  0.141860       1 14.243219
## 7: 1953 20.311890 none 2020-11-18   95.20    19.9610  0.209670       1 21.336018
## 8: 1954 5.307757 tank 2020-11-18   97.30     5.3328  0.054808       1 5.455044
## 9: 1955 17.422558 none 2020-11-18   95.20    17.1320  0.179960       1 18.301007
## 10: 1956 25.641761 none 2020-11-25   71.75    25.1850  0.351020       2 35.737646
## 11: 1957 27.487533 field 2020-11-25   72.45    26.9790  0.372390       2 37.940005
## 12: 1958 1.319052 tank 2020-11-25   67.55     1.3104  0.019399       2 1.952704
## 13: 1959 20.998197 field 2020-11-25   72.45    20.7570  0.286510       2 28.983018
## 14: 1960 1.872497 tank 2020-11-25   67.55     1.8739  0.027741       2 2.772016
## 15: 1961 25.953998 none 2020-11-25   71.75    25.3840  0.353780       2 36.172820
## 16: 1962 2.325047 tank 2020-11-25   67.55     2.3160  0.034286       2 3.441964
## 17: 1963 23.870387 field 2020-11-25   72.45    23.5660  0.325270       2 32.947393
## 18: 1964 27.513122 none 2020-11-25   71.75    26.8990  0.374900       2 38.345815
```

```

## 19: 1965 20.759140 none 2020-02-12 151.20 20.4720 0.135400 3 13.729590
## 20: 1966 3.342990 tank 2020-02-12 118.30 3.3581 0.028386 3 2.825858
## 21: 1967 17.682434 field 2020-02-12 149.10 17.5260 0.117540 3 11.859446
## 22: 1968 17.714323 field 2020-02-12 149.10 17.5560 0.117750 3 11.880834
## 23: 1969 3.174567 tank 2020-02-12 118.30 3.1914 0.026977 3 2.683489
## 24: 1970 17.449584 field 2020-02-12 149.10 17.2320 0.115580 3 11.703276
## 25: 1971 26.459916 none 2020-02-12 151.20 25.9790 0.171820 3 17.499944
## 26: 1972 3.093773 tank 2020-02-12 118.30 3.1087 0.026278 3 2.615193
## 27: 1973 24.999419 none 2020-02-12 151.20 24.6010 0.162700 3 16.534007
## 28: 1974 8.577115 tank 2020-09-12 71.40 8.6166 0.120680 3 12.012766
## 29: 1975 8.772058 tank 2020-12-09 71.40 8.8196 0.123520 4 12.285796
## 30: 1976 15.840716 field 2020-12-09 65.10 15.6990 0.241150 4 24.332897
## 31: 1977 17.579123 none 2020-09-12 66.50 17.2490 0.259380 4 26.434772
## 32: 1978 14.707340 field 2020-09-12 65.10 14.6140 0.224490 4 22.591920
## 33: 1979 19.341930 none 2020-12-09 66.50 18.9850 0.285480 4 29.085609
## 34: 1980 9.341222 tank 2020-12-09 71.40 9.3760 0.131320 4 13.082943
## 35: 1981 14.748452 field 2020-12-09 65.10 14.6650 0.225270 4 22.655071
## 36: 1982 18.794922 none 2020-12-09 66.50 18.4340 0.277210 4 28.263041
##     pmid      e_NH3 treat    app_date tan_app e_cum_final e_rel_final date_int e_NH3_rel
mod2 <- aov(e_NH3_rel ~ treat * app_date, data = amm_tot)
summary(mod2)

##                               Df Sum Sq Mean Sq F value    Pr(>F)
## treat                  2 2486.4 1243.2  400.66 < 2e-16 ***
## app_date                4 1218.1   304.5   98.14 2.82e-13 ***
## treat:app_date          8  631.5    78.9   25.44 3.88e-09 ***
## Residuals              21   65.2     3.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
drop1(mod2, test = 'F')

## Single term deletions
##
## Model:
## e_NH3_rel ~ treat * app_date
##                               Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>                      65.16  51.36
## treat:app_date               8    631.51 696.67 120.66   25.44 3.878e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
mod3 <- aov(e_rel_final ~ treat + app_date, data = amm_tot)
summary(mod3)

##                               Df Sum Sq Mean Sq F value    Pr(>F)
## treat                  2 0.23732 0.11866   51.34 2.96e-10 ***
## app_date                4 0.11738 0.02935   12.70 4.33e-06 ***
## Residuals              29 0.06702 0.00231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
coef(mod2)

##                               (Intercept)          treatattank          treatfield
##                               15.9211805           -13.2130007            -4.1066620

```

```

##          app_date2020-11-25          app_date2020-12-09  treattank:app_date2020-09-12 treatfield
##                20.8309129                  12.7531446           -1.2090051
## treatfield:app_date2020-11-18  treattank:app_date2020-11-25 treatfield:app_date2020-11-25  treattank
##                -3.0942023                 -20.8168644            0.6447074

model.tables(mod2, 'means')

## Tables of means
## Grand mean
##
## 17.09293
##
##   treat
##       none    tank field
##      25.24  5.682 20.36
## rep 12.00 12.000 12.00
##
##   app_date
## 2020-02-12 2020-09-12 2020-11-18 2020-11-25 2020-12-09
##        10.15      20.35      12.78      24.25      21.62
## rep     9.00      3.00      9.00      9.00      6.00
##
##   treat:app_date
##       app_date
## treat 2020-02-12 2020-09-12 2020-11-18 2020-11-25 2020-12-09
##   none 15.92      26.43      20.34      36.75      28.67
##   rep   3.00      1.00      3.00      3.00      2.00
##   tank  2.71      12.01      4.84      2.72      12.68
##   rep   3.00      1.00      3.00      3.00      2.00
##   field 11.81     22.59     13.14     33.29     23.49
##   rep   3.00      1.00      3.00      3.00      2.00

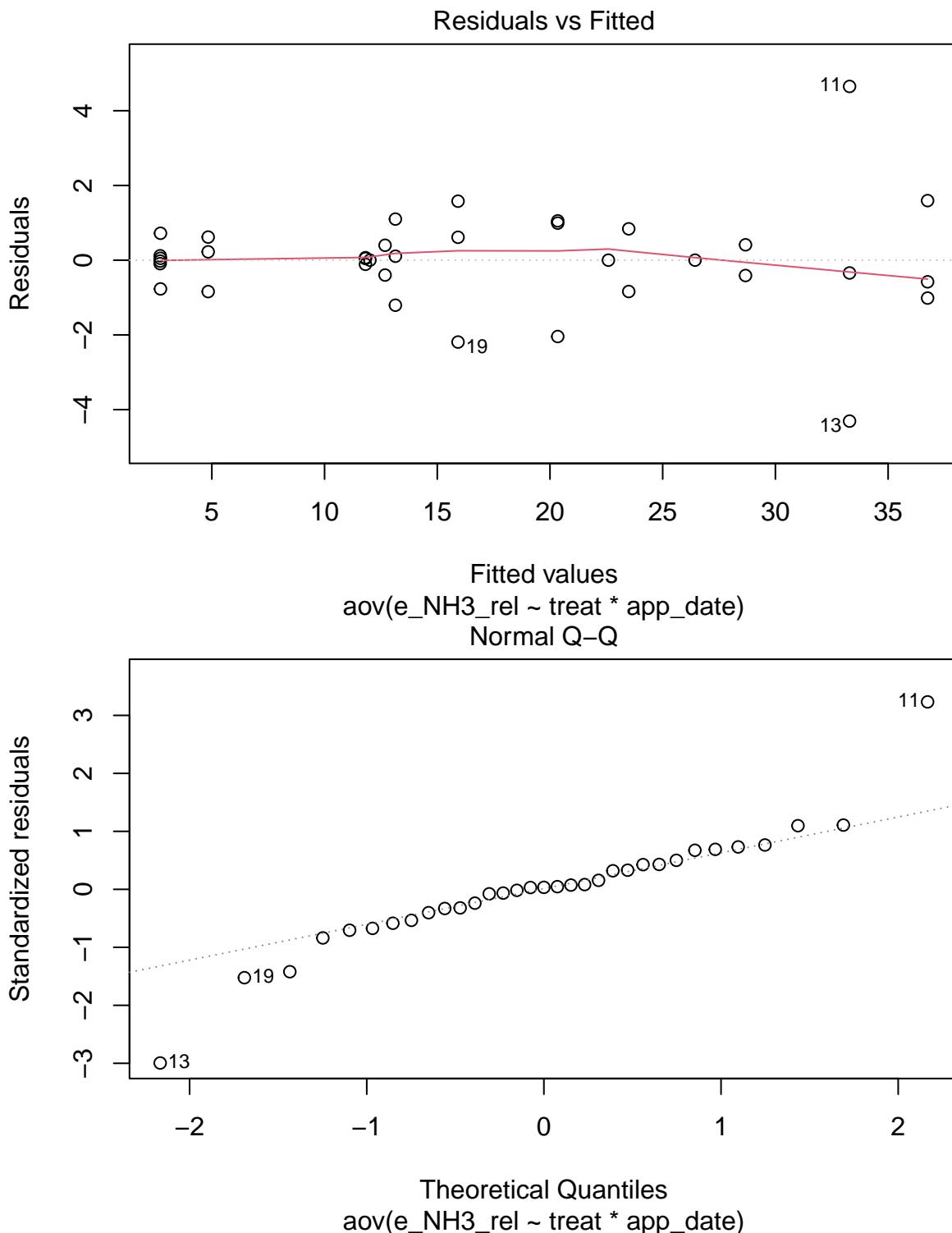
TukeyHSD(mod2, 'treat')

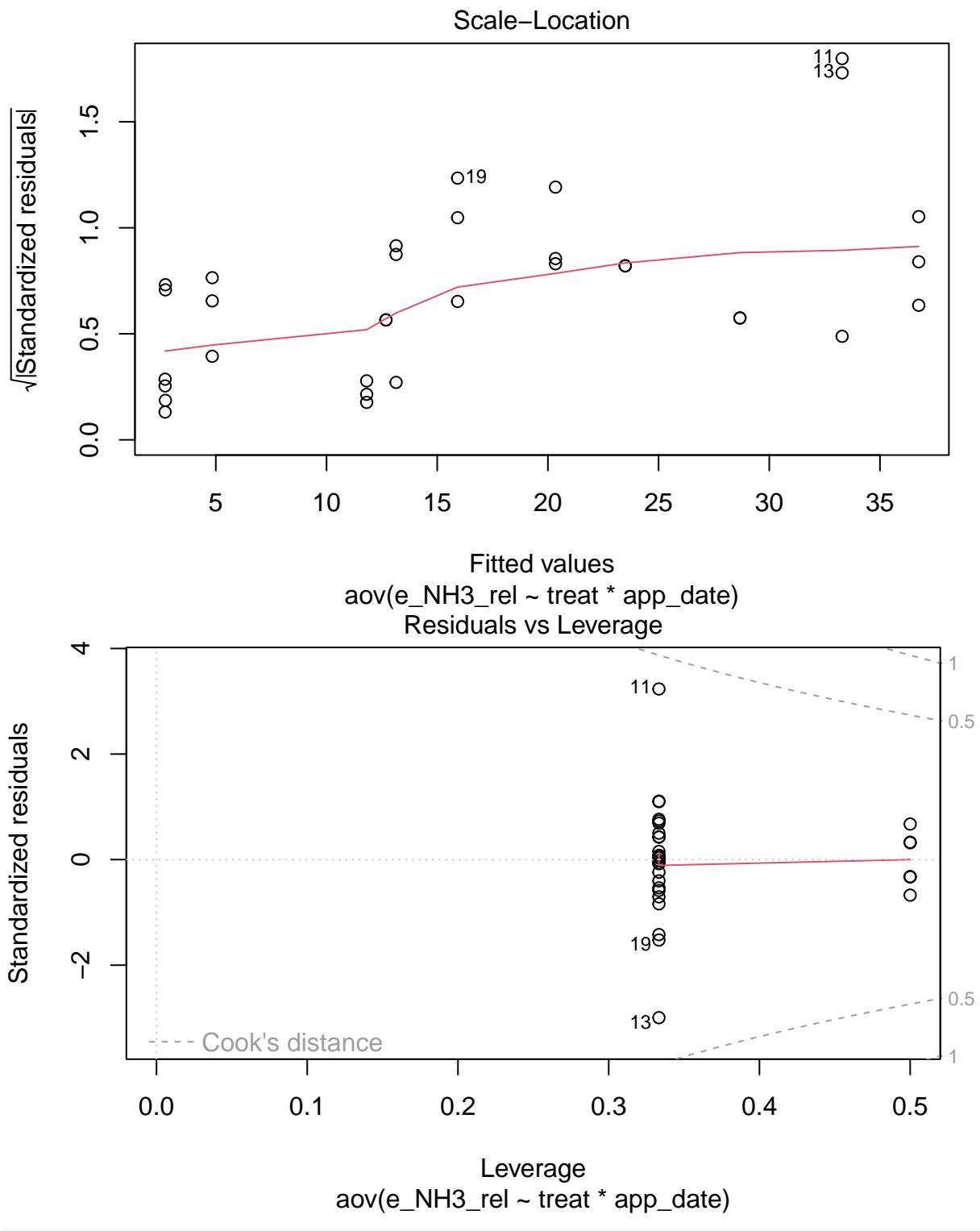
## Tukey multiple comparisons of means
## 95% family-wise confidence level
##
## Fit: aov(formula = e_NH3_rel ~ treat * app_date, data = amm_tot)
##
## $treat
##          diff      lwr      upr p adj
## tank-none -19.554307 -21.366927 -17.741687 0e+00
## field-none -4.875998 -6.688618 -3.063378 3e-06
## field-tank 14.678309 12.865689 16.490929 0e+00

plot(mod2, ask = FALSE)

## Warning: not plotting observations with leverage one:
## 28, 31, 32

```





```
confint(mod2)
```

	2.5 %	97.5 %
## (Intercept)	13.806203	18.036157
## treattank	-16.204030	-10.221971
## treatfield	-7.097691	-1.115633
## app_date2020-09-12	6.283638	14.743546

```

## app_date2020-11-18          1.432160  7.414218
## app_date2020-11-25          17.839884 23.821942
## app_date2020-12-09          9.409072 16.097217
## treattank:app_date2020-09-12 -7.191063  4.773053
## treatfield:app_date2020-09-12 -5.718249  6.245868
## treattank:app_date2020-11-18 -6.524011  1.935898
## treatfield:app_date2020-11-18 -7.324156  1.135752
## treattank:app_date2020-11-25 -25.046818 -16.586910
## treatfield:app_date2020-11-25 -3.585247  4.874661
## treattank:app_date2020-12-09 -7.506187  1.952278
## treatfield:app_date2020-12-09 -5.802911  3.655554

```

In Python. . .

```

amm_tot = amm_plot
print(amm_tot)

```

	pmid	treat	app_date	tan_app	e_cum_final	e_rel_final	date_int
## 0	1947	tank	2020-11-18	97.30	3.9108	4.0193	1
## 1	1948	tank	2020-11-18	97.30	4.9536	5.0910	1
## 2	1949	field	2020-11-18	103.60	13.6860	13.2110	1
## 3	1950	field	2020-11-18	103.60	12.3270	11.8980	1
## 4	1951	none	2020-11-18	95.20	20.0020	21.0100	1
## 5	1952	field	2020-11-18	103.60	14.6960	14.1860	1
## 6	1953	none	2020-11-18	95.20	19.9610	20.9670	1
## 7	1954	tank	2020-11-18	97.30	5.3328	5.4808	1
## 8	1955	none	2020-11-18	95.20	17.1320	17.9960	1
## 9	1956	none	2020-11-25	71.75	25.1850	35.1020	2
## 10	1957	field	2020-11-25	72.45	26.9790	37.2390	2
## 11	1958	tank	2020-11-25	67.55	1.3104	1.9399	2
## 12	1959	field	2020-11-25	72.45	20.7570	28.6510	2
## 13	1960	tank	2020-11-25	67.55	1.8739	2.7741	2
## 14	1961	none	2020-11-25	71.75	25.3840	35.3780	2
## 15	1962	tank	2020-11-25	67.55	2.3160	3.4286	2
## 16	1963	field	2020-11-25	72.45	23.5660	32.5270	2
## 17	1964	none	2020-11-25	71.75	26.8990	37.4900	2
## 18	1965	none	2020-02-12	151.20	20.4720	13.5400	3
## 19	1966	tank	2020-02-12	118.30	3.3581	2.8386	3
## 20	1967	field	2020-02-12	149.10	17.5260	11.7540	3
## 21	1968	field	2020-02-12	149.10	17.5560	11.7750	3
## 22	1969	tank	2020-02-12	118.30	3.1914	2.6977	3
## 23	1970	field	2020-02-12	149.10	17.2320	11.5580	3
## 24	1971	none	2020-02-12	151.20	25.9790	17.1820	3
## 25	1972	tank	2020-02-12	118.30	3.1087	2.6278	3
## 26	1973	none	2020-02-12	151.20	24.6010	16.2700	3
## 27	1974	tank	2020-09-12	71.40	8.6166	12.0680	3
## 28	1975	tank	2020-12-09	71.40	8.8196	12.3520	4
## 29	1976	field	2020-12-09	65.10	15.6990	24.1150	4
## 30	1977	none	2020-09-12	66.50	17.2490	25.9380	4
## 31	1978	field	2020-09-12	65.10	14.6140	22.4490	4
## 32	1979	none	2020-12-09	66.50	18.9850	28.5480	4
## 33	1980	tank	2020-12-09	71.40	9.3760	13.1320	4
## 34	1981	field	2020-12-09	65.10	14.6650	22.5270	4
## 35	1982	none	2020-12-09	66.50	18.4340	27.7210	4

```
mod2 = smf.ols('e_rel_final ~ C(treat, Treatment(reference = "none")) * app_date', amm_tot).fit()
print(mod2.summary())
```

```
##                                     OLS Regression Results
## -----
## Dep. Variable:      e_rel_final    R-squared:           0.986
## Model:              OLS            Adj. R-squared:        0.976
## Method:             Least Squares F-statistic:         103.2
## Date:               Thu, 29 Feb 2024 Prob (F-statistic):   4.02e-16
## Time:                10:38:21       Log-Likelihood:     -60.401
## No. Observations:      36            AIC:                  150.8
## Df Residuals:         21            BIC:                  174.6
## Df Model:              14
## Covariance Type:    nonrobust
## -----
##                                         coef    std err
## -----
## Intercept                               15.6640   0.979   15.99
## C(treat, Treatment(reference="none"))[T.field]      -3.9683   1.385   -2.88
## C(treat, Treatment(reference="none"))[T.tank]        -12.9426   1.385   -9.34
## app_date[T.2020-09-12]                      10.2740   1.959   5.24
## app_date[T.2020-11-18]                      4.3270   1.385   3.11
## app_date[T.2020-11-25]                      20.3260   1.385   14.67
## app_date[T.2020-12-09]                      12.4705   1.548   8.05
## C(treat, Treatment(reference="none"))[T.field]:app_date[T.2020-09-12]  0.4793   2.770   0.17
## C(treat, Treatment(reference="none"))[T.tank]:app_date[T.2020-09-12]  -0.9274   2.770   -0.33
## C(treat, Treatment(reference="none"))[T.field]:app_date[T.2020-11-18]  -2.9243   1.959   -1.48
## C(treat, Treatment(reference="none"))[T.tank]:app_date[T.2020-11-18]  -2.1847   1.959   -1.11
## C(treat, Treatment(reference="none"))[T.field]:app_date[T.2020-11-25]  0.7840   1.959   0.40
## C(treat, Treatment(reference="none"))[T.tank]:app_date[T.2020-11-25]  -20.3332   1.959   -10.34
## C(treat, Treatment(reference="none"))[T.field]:app_date[T.2020-12-09]  -0.8452   2.190   -0.38
## C(treat, Treatment(reference="none"))[T.tank]:app_date[T.2020-12-09]  -2.4499   2.190   -1.11
## -----
## Omnibus:                          10.510   Durbin-Watson:          2.239
## Prob(Omnibus):                    0.005   Jarque-Bera (JB):      27.663
## Skew:                            0.066   Prob(JB):            9.84e-07
## Kurtosis:                         7.292   Cond. No.             20.6
## -----
## 
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

NTS: FIGURE OUT CIs in Python!!!! WIP

```
mod3 = smf.ols('e_rel_final ~ C(treat, Treatment(reference = "none")) + app_date', amm_tot).fit()
mod3.summary()
```

```
## <class 'statsmodels.iolib.summary.Summary'>
## """
##                                     OLS Regression Results
## -----
## Dep. Variable:      e_rel_final    R-squared:           0.841
## Model:              OLS            Adj. R-squared:        0.808
## Method:             Least Squares F-statistic:         25.58
## Date:               Thu, 29 Feb 2024 Prob (F-statistic):   2.42e-10
```

```

## Time:          10:38:22  Log-Likelihood:      -103.72
## No. Observations:    36   AIC:                  221.4
## Df Residuals:       29   BIC:                  232.5
## Df Model:           6
## Covariance Type:   nonrobust
## -----
##                                         coef   std err      t  P>|t| [0.0]
## -----
## Intercept                         17.9143  1.963   9.128  0.000 13.9
## C(treat, Treatment(reference="none"))[T.field] -4.6043  1.963  -2.346  0.026 -8.6
## C(treat, Treatment(reference="none"))[T.tank]   -19.0577 1.963  -9.710  0.000 -23.0
## app_date[T.2020-09-12]                10.1247  3.205   3.159  0.004  3.5
## app_date[T.2020-11-18]                 2.6240  2.266   1.158  0.256 -2.0
## app_date[T.2020-11-25]                 13.8096 2.266   6.094  0.000  9.1
## app_date[T.2020-12-09]                 11.3722 2.534   4.488  0.000  6.1
## -----
## Omnibus:                      3.175 Durbin-Watson:        2.799
## Prob(Omnibus):                0.204 Jarque-Bera (JB):     1.922
## Skew:                          -0.478 Prob(JB):          0.383
## Kurtosis:                     3.606 Cond. No.            6.10
## -----
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## """

```

Mixed-effects model

Mixed-effects models can be used for some really powerful stuff, but there is not always a clear advantage in understanding. Here we can use it to handle the data structure in an arguably more appropriate way.

```
library(lme4)
```

```
me1 <- lmer(e_NH3_rel ~ treat + (1|app_date), data = amm_tot)
```

```
summary(me1)
```

```

## Linear mixed model fit by REML ['lmerMod']
## Formula: e_NH3_rel ~ treat + (1 | app_date)
##   Data: amm_tot
##
## REML criterion at convergence: 215.3
##
## Scaled residuals:
##       Min     1Q   Median     3Q    Max
## -2.13052 -0.44537 -0.04327  0.68609  2.22431
##
## Random effects:
##   Groups   Name        Variance Std.Dev.
##   app_date (Intercept) 34.54     5.877
##   Residual             23.94     4.893
## Number of obs: 36, groups:  app_date, 5
##
## Fixed effects:
##   Estimate Std. Error t value
## (Intercept)  25.880     3.002   8.619

```

```

## treatattank -19.554      1.998 -9.789
## treatfield   -4.876      1.998 -2.441
##
## Correlation of Fixed Effects:
##           (Intr) trttnk
## treatattank -0.333
## treatfield  -0.333  0.500
confint(me2)

## Computing profile confidence intervals ...

##          2.5 %    97.5 %
## .sig01     4.5401234 15.6626901
## .sig02    -0.9708568 -0.2114638
## .sig03    -1.0000000  1.0000000
## .sig04     4.3864961 16.7584945
## .sig05    -1.0000000  1.0000000
## .sig06    0.0000000  2.7319728
## .sigma     1.3143052  2.2738549
## (Intercept) 17.9778363 33.4768466
## treatattank -27.1421534 -10.4970476
## treatfield   -6.7408736 -3.2454944

me2 <- lmer(e_NH3_rel ~ treat + (treat|app_date), data = amm_tot)
summary(me2)

## Linear mixed model fit by REML ['lmerMod']
## Formula: e_NH3_rel ~ treat + (treat | app_date)
##   Data: amm_tot
##
## REML criterion at convergence: 166.9
##
## Scaled residuals:
##       Min     1Q   Median     3Q    Max
## -2.44089 -0.42677 -0.00649  0.37310  2.76882
##
## Random effects:
##   Groups   Name        Variance Std.Dev. Corr
##   app_date (Intercept) 63.5789  7.9736
##             treatattank  72.2202  8.4982  -0.83
##             treatfield   0.6801  0.8247   0.86 -1.00
##   Residual            2.9559  1.7193
## Number of obs: 36, groups:  app_date, 5
##
## Fixed effects:
##           Estimate Std. Error t value
## (Intercept) 25.7178    3.6034   7.137
## treatattank -18.8330    3.8728  -4.863
## treatfield   -4.9371    0.7932  -6.224
##
## Correlation of Fixed Effects:
##           (Intr) trttnk
## treatattank -0.829
## treatfield   0.309 -0.378

```

```

anova(me2, me1, test = 'Chisquare')

## refitting model(s) with ML (instead of REML)

## Data: amm_tot
## Models:
## me1: e_NH3_rel ~ treat + (1 | app_date)
## me2: e_NH3_rel ~ treat + (treat | app_date)
##      npar    AIC    BIC  logLik deviance Chisq Df Pr(>Chisq)
## me1     5 235.16 243.08 -112.580   225.16
## me2    10 195.64 211.48  -87.822   175.64 49.517  5  1.74e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
AIC(me2, me1)

##      df      AIC
## me2 10 186.8874
## me1  5 225.3088

coef(me1)

## $app_date
##             (Intercept) treatattank treatfield
## 2020-02-12    18.83412 -19.55431  -4.875998
## 2020-09-12    27.99994 -19.55431  -4.875998
## 2020-11-18    21.27333 -19.55431  -4.875998
## 2020-11-25    31.93203 -19.55431  -4.875998
## 2020-12-09    29.35897 -19.55431  -4.875998
##
## attr(),"class"
## [1] "coef.mer"

coef(me2)

## $app_date
##             (Intercept) treatattank treatfield
## 2020-02-12    16.68873 -13.89552  -5.459408
## 2020-09-12    27.14535 -15.77183  -5.201567
## 2020-11-18    19.43713 -14.46361  -5.384870
## 2020-11-25    36.67792 -33.73477  -3.498317
## 2020-12-09    28.63982 -16.29907  -5.141313
##
## attr(),"class"
## [1] "coef.mer"

coef(mod3)

##             (Intercept)          treatattank          treatfield
## 0.17914350        -0.19057683        -0.04604333        0.10124656        0.02624000

coef(mod2)

##             (Intercept)          treatattank          treatfield
## 15.9211805       -13.2130007       -4.1066620
## app_date2020-11-25 app_date2020-12-09 treatattank:app_date2020-09-12 treatfield
## 20.8309129        12.7531446       -1.2090051
## treatfield:app_date2020-11-18 treatattank:app_date2020-11-25 treatfield:app_date2020-11-25 treattank
```

```

## -3.0942023 -20.8168644 0.6447074
coef(me1)

## $app_date
## (Intercept) treatattank treatfield
## 2020-02-12 18.83412 -19.55431 -4.875998
## 2020-09-12 27.99994 -19.55431 -4.875998
## 2020-11-18 21.27333 -19.55431 -4.875998
## 2020-11-25 31.93203 -19.55431 -4.875998
## 2020-12-09 29.35897 -19.55431 -4.875998
##
## attr(),"class")
## [1] "coef.mer"
confint(me1, parm = c('treatattank', 'treatfield'))

## Computing profile confidence intervals ...
## 2.5 % 97.5 %
## treatattank -23.458561 -15.6500534
## treatfield -8.780252 -0.9717443
coef(me2)

## $app_date
## (Intercept) treatattank treatfield
## 2020-02-12 16.68873 -13.89552 -5.459408
## 2020-09-12 27.14535 -15.77183 -5.201567
## 2020-11-18 19.43713 -14.46361 -5.384870
## 2020-11-25 36.67792 -33.73477 -3.498317
## 2020-12-09 28.63982 -16.29907 -5.141313
##
## attr(),"class")
## [1] "coef.mer"
confint(me2, parm = c('treatattank', 'treatfield'))

## Computing profile confidence intervals ...
## 2.5 % 97.5 %
## treatattank -27.142153 -10.497048
## treatfield -6.740874 -3.245494

```

In Python, categorical variables are trickier to handle.

```

print(ammm_tot)

##   pmid treat app_date tan_app e_cum_final e_rel_final date_int
## 0  1947 tank 2020-11-18    97.30     3.9108    4.0193      1
## 1  1948 tank 2020-11-18    97.30     4.9536    5.0910      1
## 2  1949 field 2020-11-18   103.60    13.6860   13.2110      1
## 3  1950 field 2020-11-18   103.60    12.3270   11.8980      1
## 4  1951 none  2020-11-18    95.20    20.0020   21.0100      1
## 5  1952 field 2020-11-18   103.60    14.6960   14.1860      1
## 6  1953 none  2020-11-18    95.20    19.9610   20.9670      1
## 7  1954 tank  2020-11-18    97.30     5.3328    5.4808      1
## 8  1955 none  2020-11-18    95.20    17.1320   17.9960      1
## 9  1956 none  2020-11-25    71.75    25.1850   35.1020      2

```

```

## 10 1957 field 2020-11-25 72.45 26.9790 37.2390 2
## 11 1958 tank 2020-11-25 67.55 1.3104 1.9399 2
## 12 1959 field 2020-11-25 72.45 20.7570 28.6510 2
## 13 1960 tank 2020-11-25 67.55 1.8739 2.7741 2
## 14 1961 none 2020-11-25 71.75 25.3840 35.3780 2
## 15 1962 tank 2020-11-25 67.55 2.3160 3.4286 2
## 16 1963 field 2020-11-25 72.45 23.5660 32.5270 2
## 17 1964 none 2020-11-25 71.75 26.8990 37.4900 2
## 18 1965 none 2020-02-12 151.20 20.4720 13.5400 3
## 19 1966 tank 2020-02-12 118.30 3.3581 2.8386 3
## 20 1967 field 2020-02-12 149.10 17.5260 11.7540 3
## 21 1968 field 2020-02-12 149.10 17.5560 11.7750 3
## 22 1969 tank 2020-02-12 118.30 3.1914 2.6977 3
## 23 1970 field 2020-02-12 149.10 17.2320 11.5580 3
## 24 1971 none 2020-02-12 151.20 25.9790 17.1820 3
## 25 1972 tank 2020-02-12 118.30 3.1087 2.6278 3
## 26 1973 none 2020-02-12 151.20 24.6010 16.2700 3
## 27 1974 tank 2020-09-12 71.40 8.6166 12.0680 3
## 28 1975 tank 2020-12-09 71.40 8.8196 12.3520 4
## 29 1976 field 2020-12-09 65.10 15.6990 24.1150 4
## 30 1977 none 2020-09-12 66.50 17.2490 25.9380 4
## 31 1978 field 2020-09-12 65.10 14.6140 22.4490 4
## 32 1979 none 2020-12-09 66.50 18.9850 28.5480 4
## 33 1980 tank 2020-12-09 71.40 9.3760 13.1320 4
## 34 1981 field 2020-12-09 65.10 14.6650 22.5270 4
## 35 1982 none 2020-12-09 66.50 18.4340 27.7210 4

me1 = smf.mixedlm('e_rel_final ~ C(treat, Treatment(reference = "none")) + (1|date_int)', amm_tot, group
## /home/sasha/.local/share/r-miniconda/envs/r-reticulate/lib/python3.6/site-packages/statsmodels/regre
## warnings.warn(msg, ConvergenceWarning)
print(me1.summary())

##                                     Mixed Linear Model Regression Results
## =====
## Model:                          MixedLM      Dependent Variable:    e_rel_final
## No. Observations:                 36          Method:                  REML
## No. Groups:                      36          Scale:                   24.4632
## Min. group size:                  1          Log-Likelihood:        -113.4857
## Max. group size:                  1          Converged:                Yes
## Mean group size:                 1.0
## -----
##                                         Coef.  Std.Err.     z   P>|z| [0.025  0.975]
## -----
## Intercept                           18.677   2.252   8.293  0.000  14.263  23.091
## C(treat, Treatment(reference="none"))[T.field] -4.604   2.855  -1.612  0.107 -10.201  0.992
## C(treat, Treatment(reference="none"))[T.tank]  -18.720   2.844  -6.583  0.000 -24.293 -13.146
## 1 | date_int                           2.028   0.158  12.801  0.000   1.718   2.339
## Group Var                            24.463
## =====
```

Including variation in treatment effects among dates is also tricky.

```
#me2 = smf.mixedlm('e_rel_final ~ C(treat, Treatment(reference = "none")) + (1/f) + (C(treat)/f)', amm_
#print(me1.summary())
```

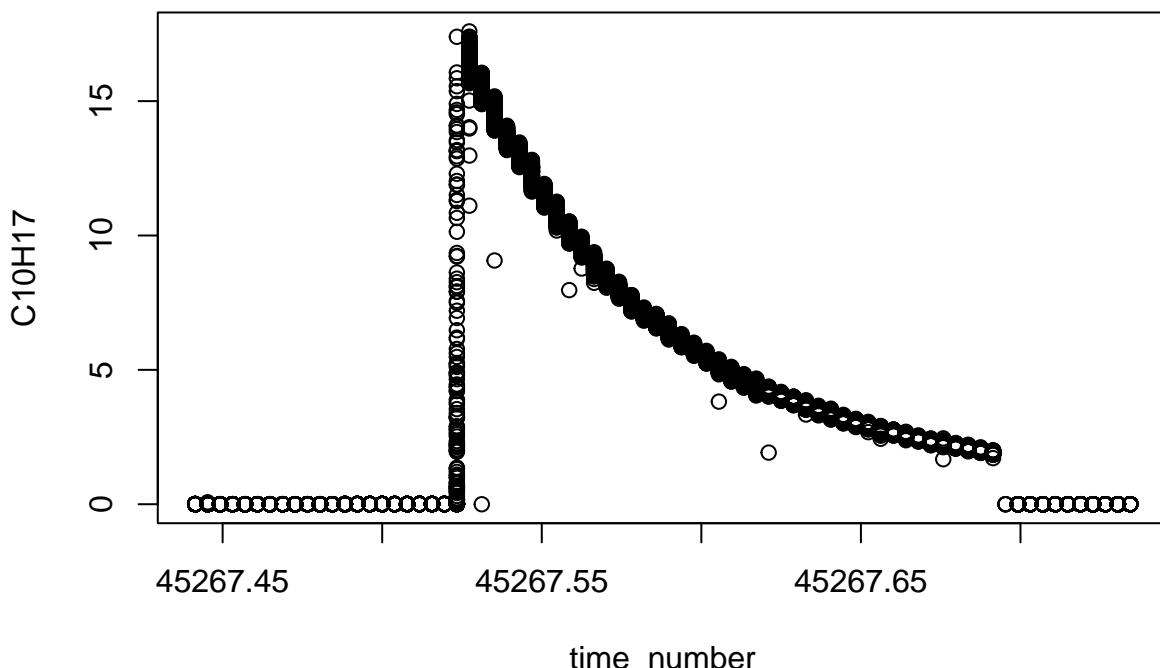
3. Nonlinear regression

Here we will use nonlinear regression to extract a rate constant from reaction chamber measurements.

```
ox <- fread('../data/VOC_reaction.csv', skip = 2)
ox
```

```
##           time_string time_number      C1H302      C3H701      C2H502      C7H1102      C10H17      C9H150
## 1: 12/7/2023 10:34     45267.44 0.26983000 0.23759000 0.19983000 0.00494390 4.641300e-03 0.032477
## 2: 12/7/2023 10:34     45267.44 0.26303000 0.25205000 0.18137000 0.00074215 1.215800e-03 0.035656
## 3: 12/7/2023 10:34     45267.44 0.27097000 0.22796000 0.19361000 0.00399780 2.126600e-03 0.038066
## 4: 12/7/2023 10:34     45267.44 0.24479000 0.19712000 0.17835000 0.00546340 2.674500e-03 0.038752
## 5: 12/7/2023 10:34     45267.44 0.28258000 0.23840000 0.18143000 0.00438240 5.596100e-03 0.036147
##   ---
## 12733: 12/7/2023 17:38     45267.73 0.03405808 0.02747731 0.01098923 0.01535654 1.477731e-03 0.128273
## 12734: 12/7/2023 17:38     45267.73 0.03853077 0.02504192 0.01067731 0.01657769 6.119231e-04 0.130203
## 12735: 12/7/2023 17:38     45267.73 0.03404269 0.02224269 0.01141692 0.01477038 1.053115e-03 0.129607
## 12736: 12/7/2023 17:38     45267.73 0.03497692 0.02501538 0.01088462 0.01620885 8.473462e-04 0.128519
## 12737: 12/7/2023 17:38     45267.73 0.03452077 0.02439692 0.01039846 0.01530308 1.123654e-05 0.126469
##           C8H1304      C9H1504      C10H1704
## 1: -0.000329700 0.000333540 0.0010117000
## 2: 0.000026400 0.000302220 -0.0008735800
## 3: 0.000372220 0.000087300 -0.0004013000
## 4: -0.000260980 -0.000734150 0.0007662200
## 5: -0.000741000 0.000602880 0.0007395400
##   ---
## 12733: 0.001797462 0.001629615 0.0013644231
## 12734: 0.002281692 0.001849769 0.0009983462
## 12735: 0.001946538 0.001774385 0.0013291923
## 12736: 0.002222885 0.001807192 0.0008379615
## 12737: 0.001943769 0.002313885 0.0008159615
```

```
plot(C10H17 ~ time_number, data = ox)
```



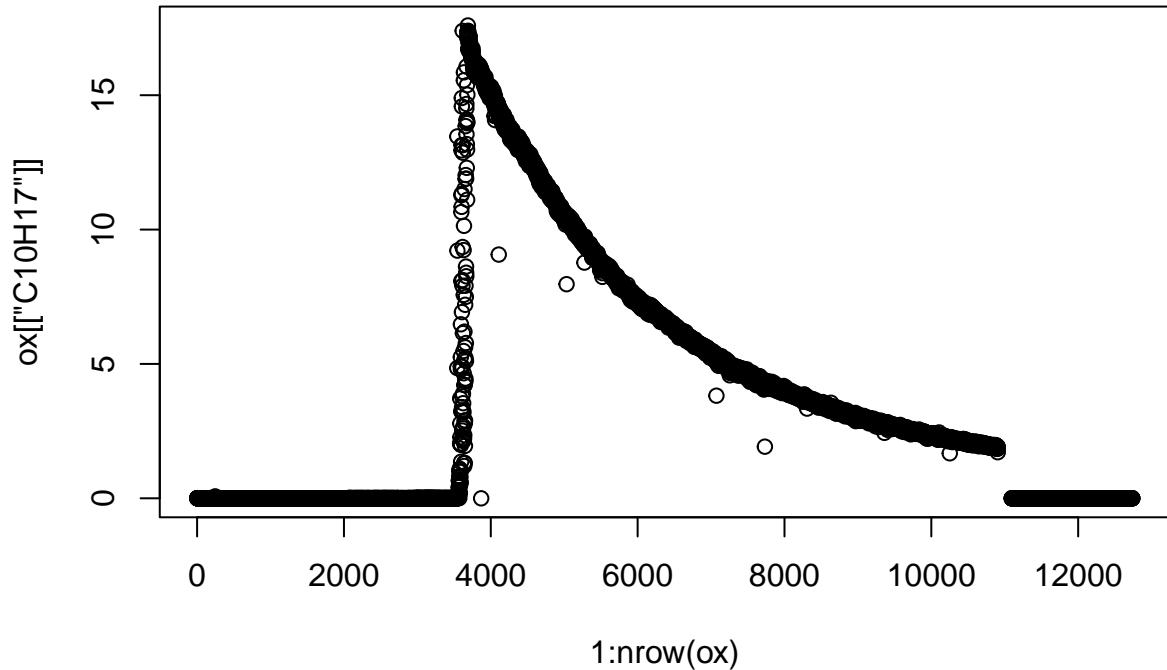
```

head(order(-ox[, C10H17]))

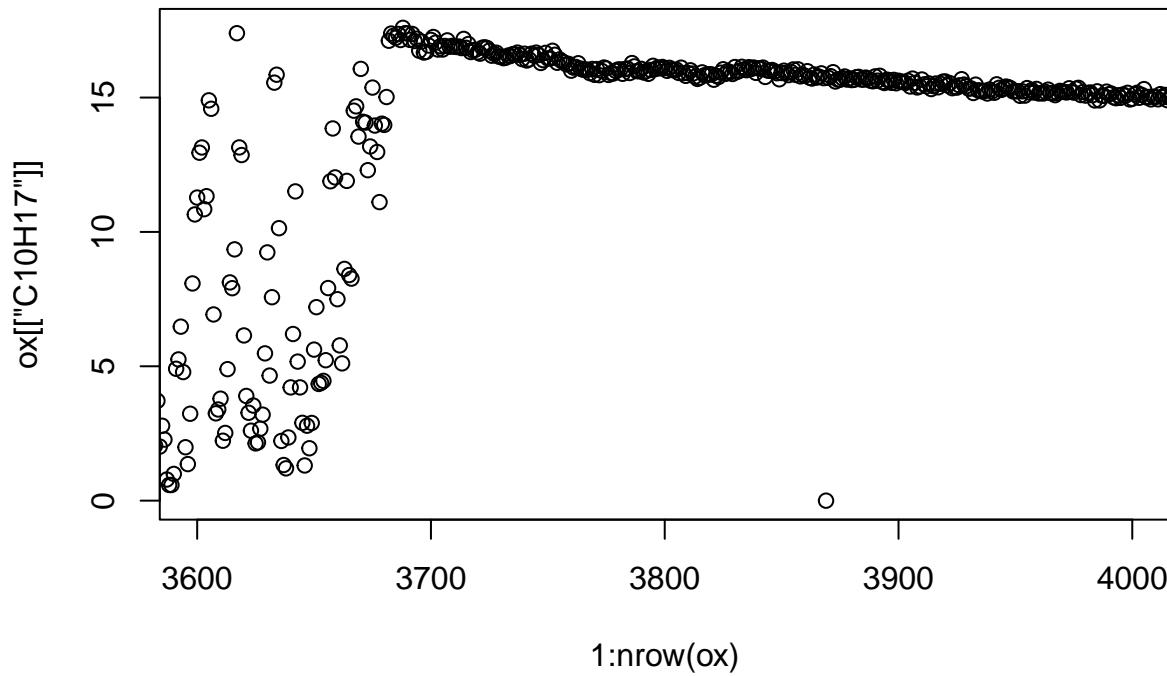
## [1] 3688 3689 3617 3690 3683 3692
ox[3688, time_number]

## [1] 45267.53
plot(1:nrow(ox), ox[['C10H17']])

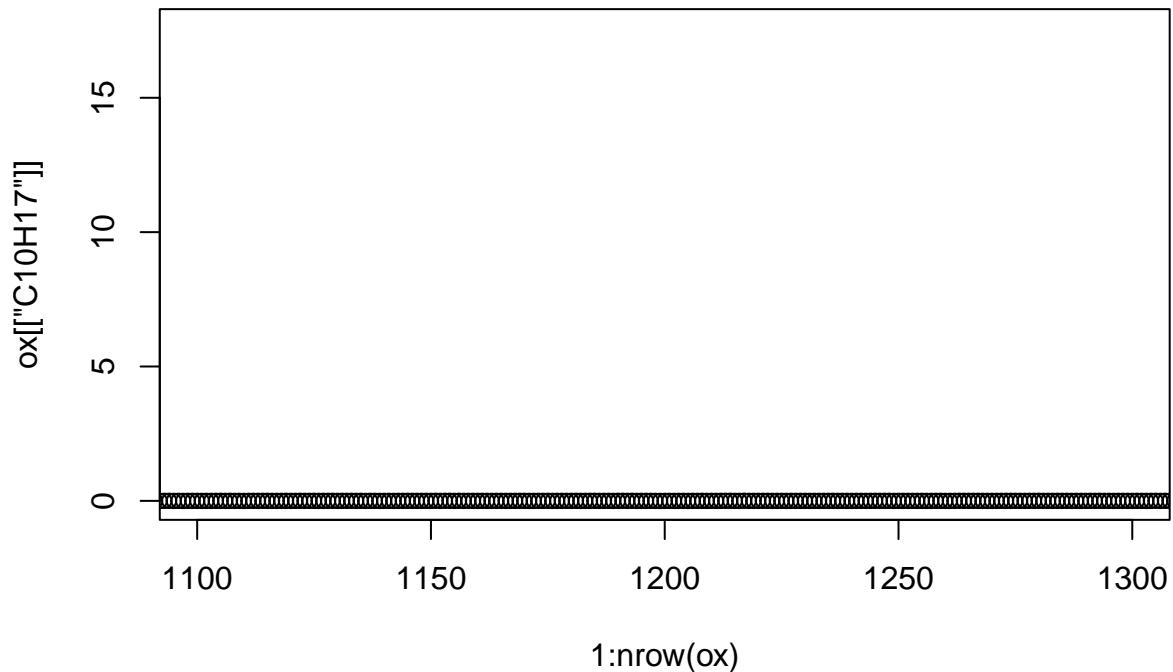
```



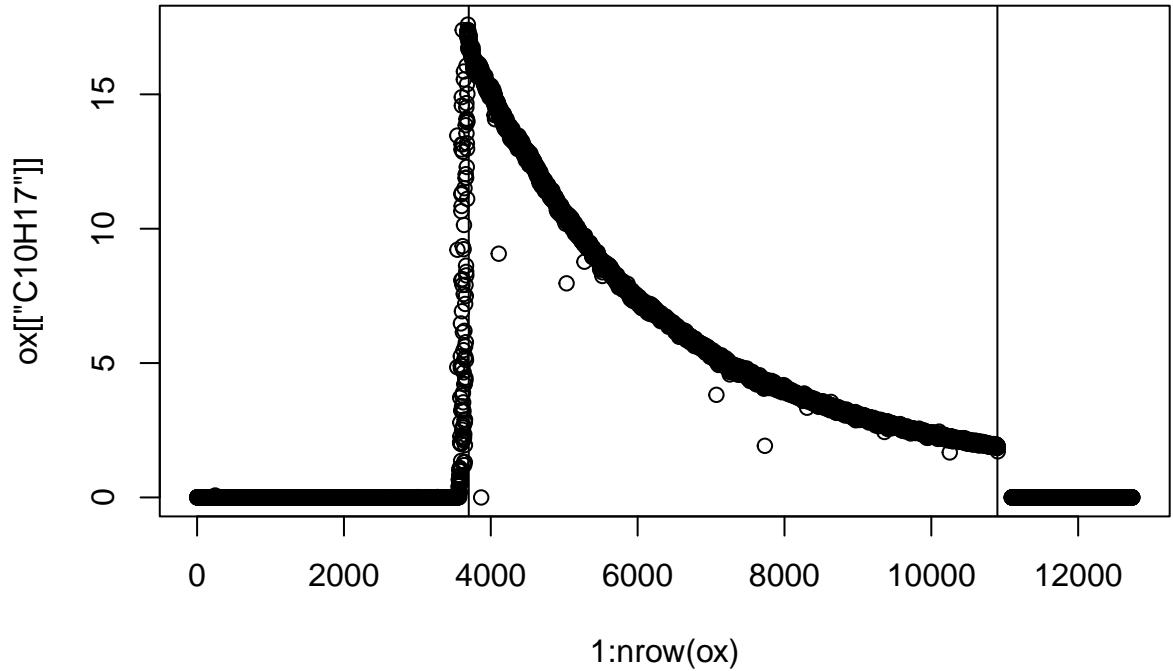
```
plot(1:nrow(ox), ox[['C10H17']], xlim = c(3600, 4000))
```



```
plot(1:nrow(ox), ox[['C10H17']], xlim = c(1100, 1300))
```

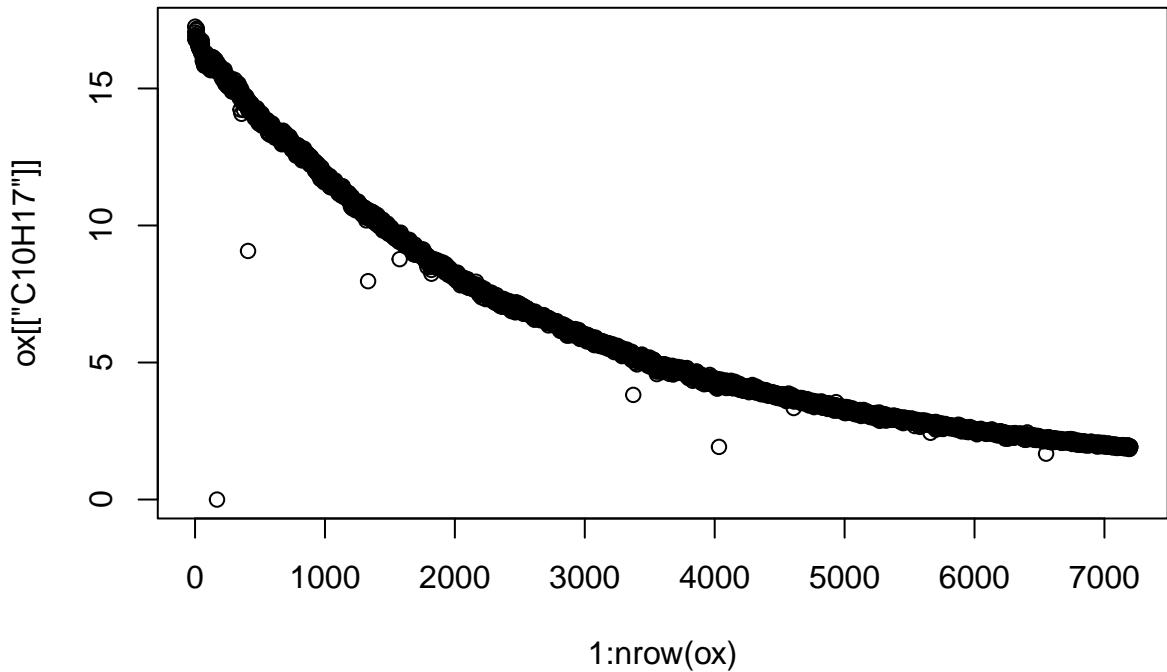


```
plot(1:nrow(ox), ox[['C10H17']])
abline(v = 3700)
abline(v = 10900)
```



```
ox <- ox[c(-1:-3700, -10900:-nrow(ox)), ]
```

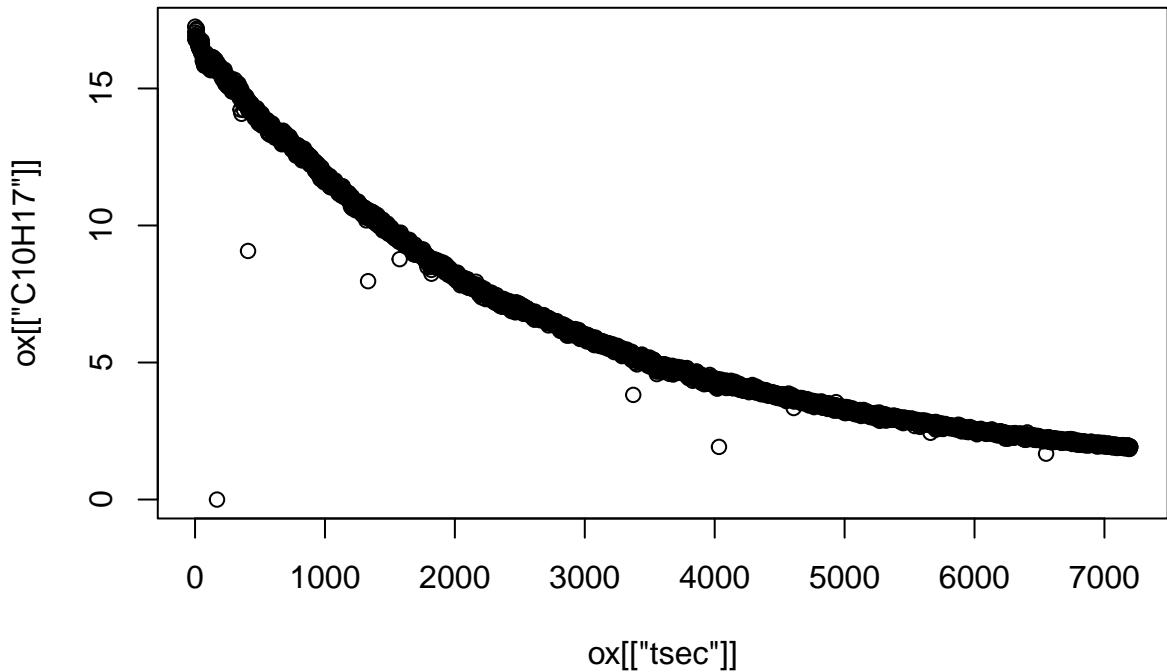
```
plot(1:nrow(ox), ox[['C10H17']])
```



```
ox[, tsec := 1:nrow(ox)]
ox
```

```
##           time_string time_number  C1H302  C3H701  C2H502  C7H1102  C10H17 C9H1501  C8H1502 C9H1502
## 1: 12/7/2023 12:37      45267.53 0.48431 0.67633 0.27782 0.014327 17.2541 0.58644 0.0068041 0.15859
## 2: 12/7/2023 12:37      45267.53 0.53386 0.64853 0.27303 0.015742 17.0439 0.64274 0.0075032 0.14346
## 3: 12/7/2023 12:37      45267.53 0.49192 0.65911 0.26383 0.014787 16.7865 0.60899 0.0085258 0.16083
## 4: 12/7/2023 12:37      45267.53 0.51608 0.61097 0.27346 0.014162 16.9080 0.62952 0.0058579 0.14947
## 5: 12/7/2023 12:37      45267.53 0.53738 0.66043 0.26720 0.012270 16.7876 0.62864 0.0015246 0.15648
##   ---
## 7195: 12/7/2023 16:37      45267.69 1.32770 3.08730 0.57081 0.132810 1.9442 6.02930 0.0122850 1.80000
## 7196: 12/7/2023 16:37      45267.69 1.28920 2.99580 0.59899 0.117230 1.9277 6.01140 0.0145900 1.82750
## 7197: 12/7/2023 16:37      45267.69 1.25940 3.12300 0.56485 0.136080 1.9304 6.00870 0.0220580 1.81800
## 7198: 12/7/2023 16:37      45267.69 1.29540 3.07490 0.54210 0.122420 1.9194 5.98660 0.0193720 1.85040
## 7199: 12/7/2023 16:37      45267.69 1.33740 3.08580 0.56955 0.113580 1.9119 5.98520 0.0157170 1.83640
##           tsec
## 1:     1
## 2:     2
## 3:     3
## 4:     4
## 5:     5
##   ---
## 7195: 7195
## 7196: 7196
## 7197: 7197
## 7198: 7198
## 7199: 7199
```

```
plot(ox[['tsec']], ox[['C10H17']])
```



```

args(nls)

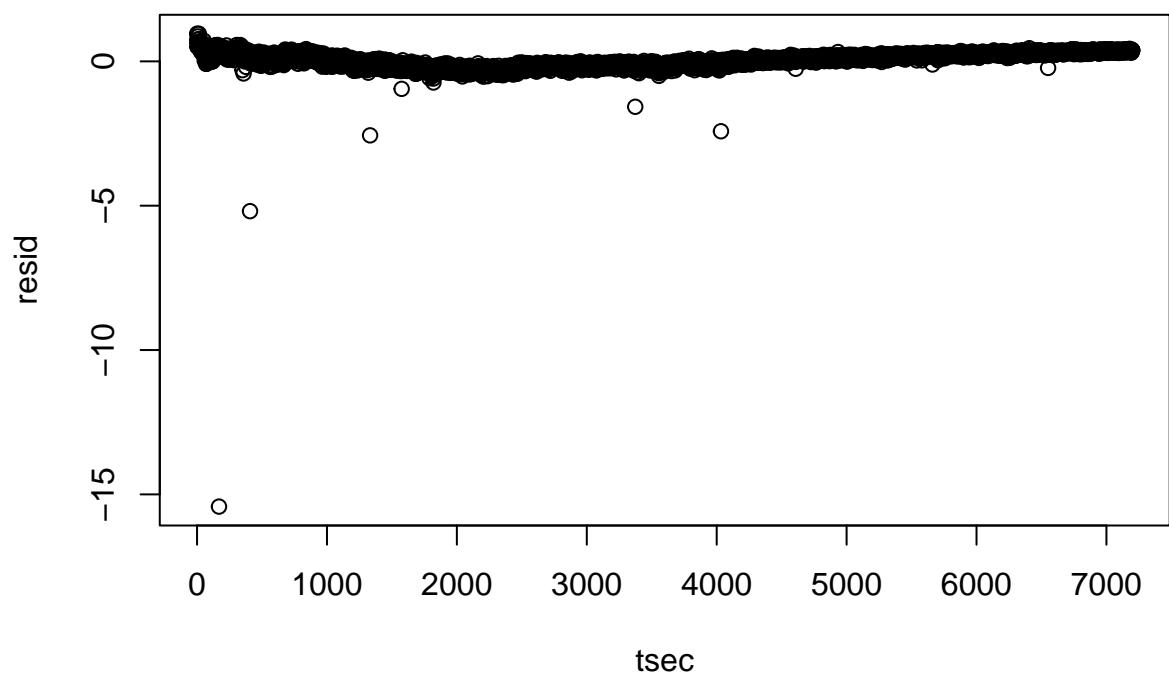
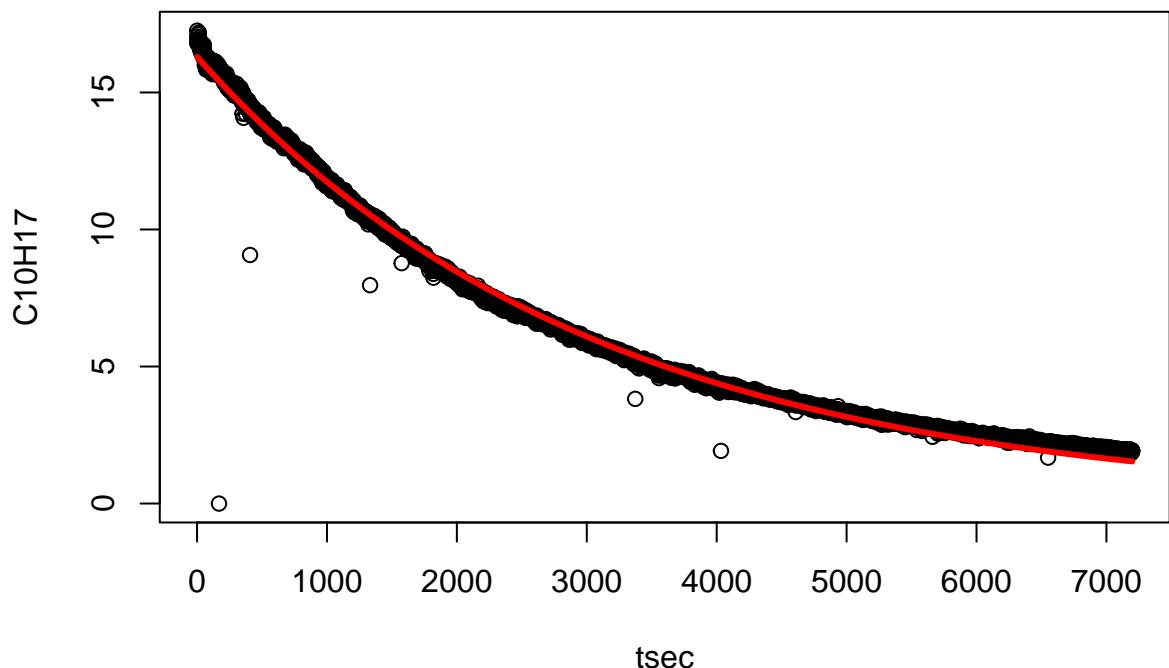
## function (formula, data = parent.frame(), start, control = nls.control(),
##   algorithm = c("default", "plinear", "port"), trace = FALSE,
##   subset, weights, na.action, model = FALSE, lower = -Inf,
##   upper = Inf, ...)
## NULL

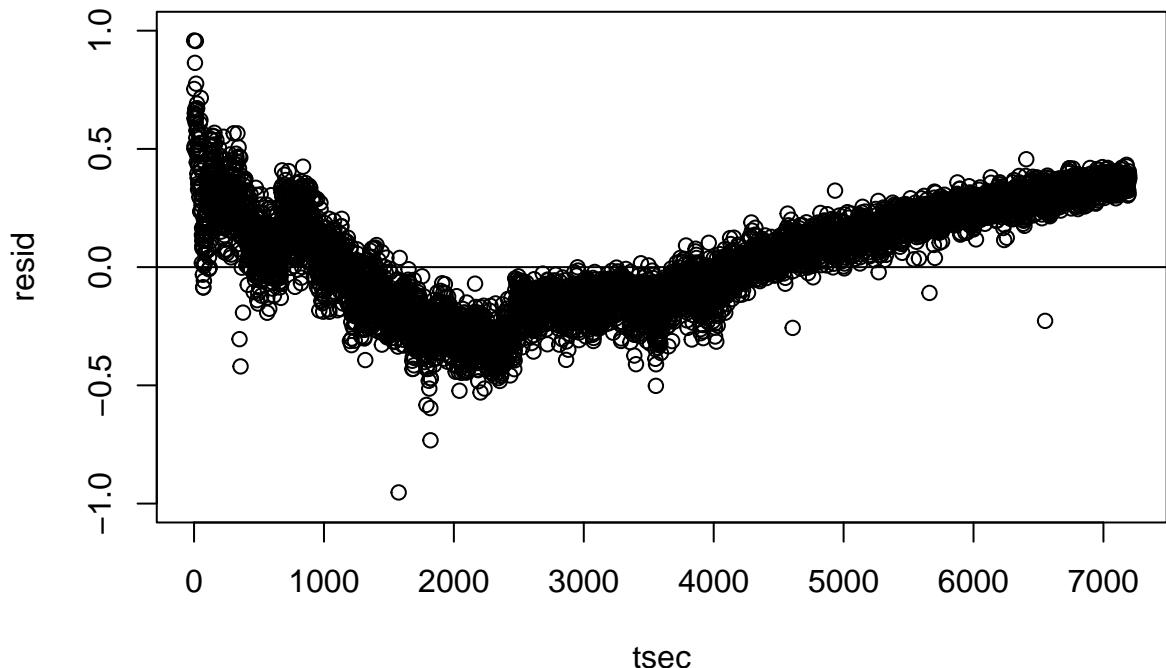
mod1 <- nls(C10H17 ~ c0 * exp(-k * tsec), start = c(c0 = 16, k = 1/4000), data = ox)
summary(mod1)

##
## Formula: C10H17 ~ c0 * exp(-k * tsec)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
##   c0 1.630e+01 1.126e-02 1448.3  <2e-16 ***
##   k  3.280e-04 3.458e-07  948.5  <2e-16 ***
##   ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2983 on 7197 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 1.485e-06

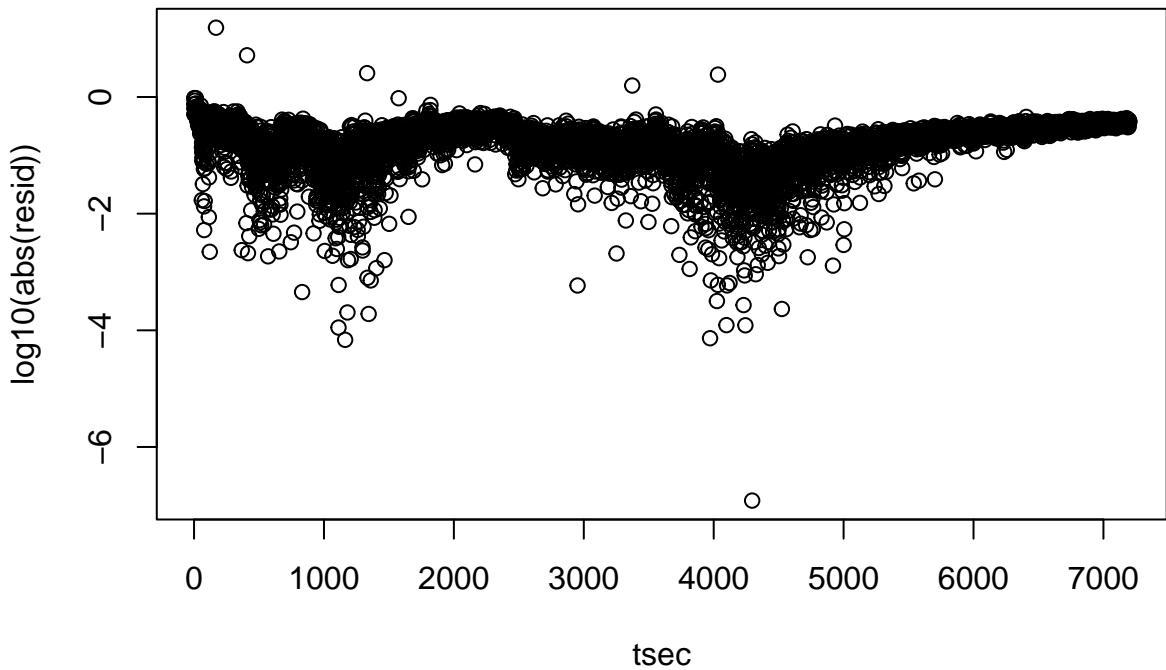
ox[, conc_pred := predict(mod1)]
ox[, resid := resid(mod1)]
plot(C10H17 ~ tsec, data = ox)
lines(conc_pred ~ tsec, data = ox, col = 'red', lwd = 3)

```





```
plot(log10(abs(resid)) ~ tsec, data = ox)
```



```
ox <- ox[abs(resid) < 1, ]
```

```
mod2 <- nls(C10H17 ~ c0 * exp(-k * tsec), start = c(c0 = 16, k = 1/4000), data = ox)
summary(mod2)
```

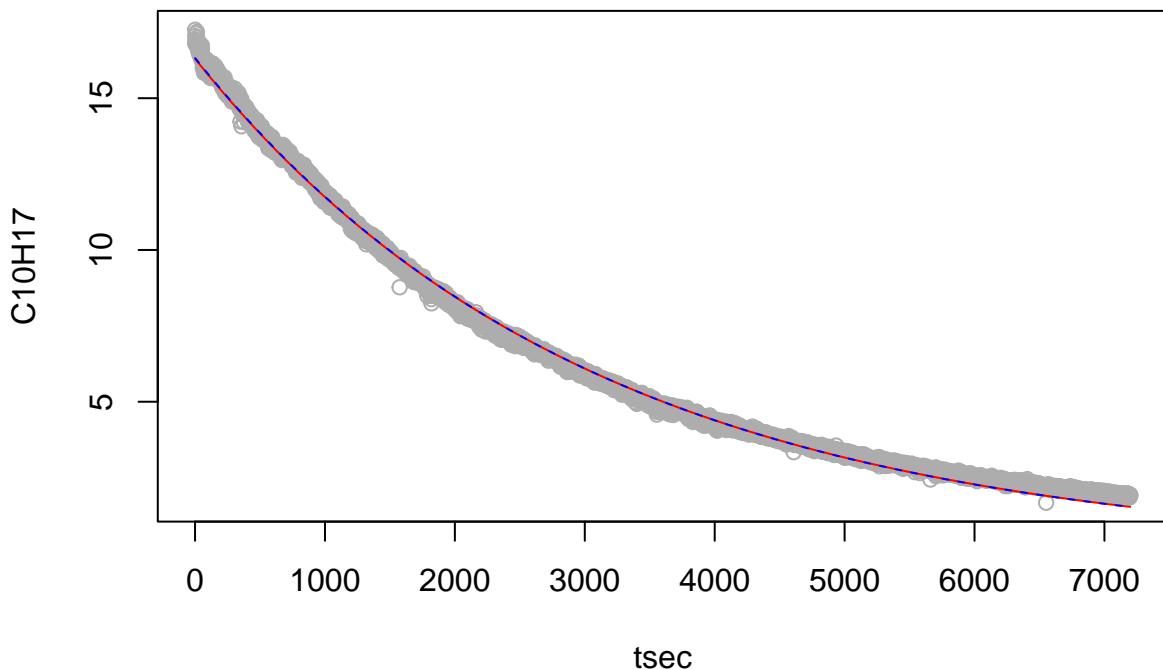
```
##
## Formula: C10H17 ~ c0 * exp(-k * tsec)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
```

```

## c0 1.633e+01 8.458e-03    1930    <2e-16 ***
## k  3.285e-04  2.596e-07    1265    <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2238 on 7192 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 2.163e-06
ox[, conc_pred2 := predict(mod2)]
ox[, resid2 := resid(mod2)]

plot(C10H17 ~ tsec, data = ox, col = 'gray68')
lines(conc_pred ~ tsec, data = ox, col = 'red')
lines(conc_pred2 ~ tsec, data = ox, col = 'blue', lty = 2)

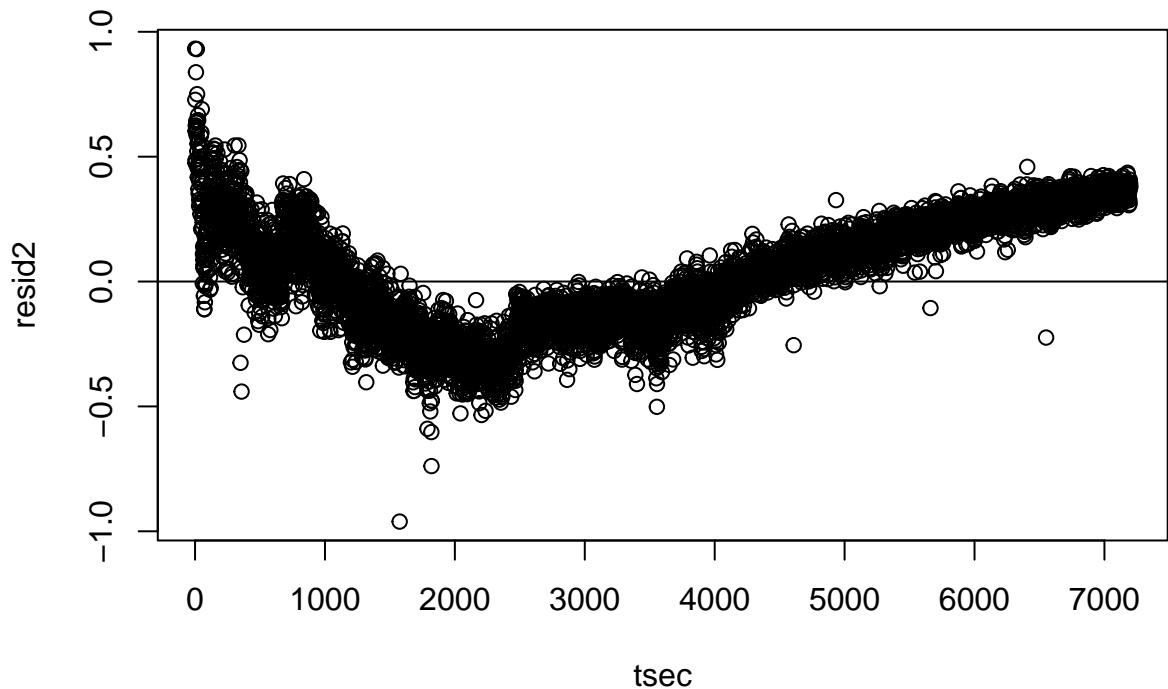
```



```

plot(resid2 ~ tsec, data = ox)
abline(h = 0)

```



NTS: SORT OUT nls in Python WIP