Exploration of rate constant/time substitution

Sasha D. Hafner

10 September, 2024 Sep:09

Packages

```r
library(data.table)
library(knitr)
library(ALFAM2)
library(ggplot2)
```

```r
packageVersion('ALFAM2')
```

```
## [1] '4.1.3'
```

Setup

Parameters. Mitigation (0), reference (1), and doubled (2) r1 and r3.

```r
print(ALFAM2::alfam2pars03)
```

```
##          int.f0    app.mthd.os.f0    app.mthd.cs.f0 man.source.pig.f0
##      0.43613933       -2.93492578       -7.80196997       -0.85171386
##       man.dm.f0            int.r1    app.mthd.bc.r1    app.mthd.ts.r1
##      0.49659337       -1.46760800        0.71991146       -0.09333684
##       man.dm.r1         man.ph.r1       air.temp.r1      wind.sqrt.r1
##     -0.02843126        0.44886708        0.03454900        0.46628989
##          int.r2      rain.rate.r2            int.r3    app.mthd.cs.r3
##     -1.20493824        0.62051420       -2.71593590       -0.34883867
##  incorp.deep.r3         man.ph.r3 incorp.shallow.f4    incorp.deep.f4
##     -1.96259695        0.03557064       -1.37979544       -3.26822034
```

```
##             int.r5      rain.rate.r5
##        -1.80000000        0.34944126
```

```
p0 <- c(int.f0 = 0.4, int.r1 = -1.5 - 0.3, int.r2 = -1.2, int.r3 = -2.7 - 0.3, int.r5 = -1.8)
p1 <- c(int.f0 = 0.4, int.r1 = -1.5, int.r2 = -1.2, int.r3 = -2.7, int.r5 = -1.8)
p2 <- c(int.f0 = 0.4, int.r1 = -1.5 + 0.3, int.r2 = -1.2, int.r3 = -2.7 + 0.3, int.r5 = -1.8)
```

Input data.

```
dat <- data.table(ct = c(2, 4, 8) * 24, TAN.app = 100)
```

Predictions

```
pred0 <- alfam2(dat, pars = p0)
```

```
## User-supplied parameters are being used.
```

```
## Warning in prepDat(dat, warn = warn): Argument prep.dum = TRUE but there are no variables to convert to dummy variables!
##   Ignoring prep.dum = TRUE.
```

```
pred1 <- alfam2(dat, pars = p1)
```

```
## User-supplied parameters are being used.
```

```
## Warning in prepDat(dat, warn = warn): Argument prep.dum = TRUE but there are no variables to convert to dummy variables!
##   Ignoring prep.dum = TRUE.
```

```
pred2 <- alfam2(dat, pars = p2)
```

```
## User-supplied parameters are being used.
```

```
## Warning in prepDat(dat, warn = warn): Argument prep.dum = TRUE but there are no variables to convert to dummy variables!
##   Ignoring prep.dum = TRUE.
```

Doubling pars effect:

```
(pred2 / pred1)[, 'er'] - 1
```

```
## [1] 0.5542342 0.5382865 0.5261679
```

Halving:

```
1 - (pred0 / pred1)[, 'er']
```

```
## [1] 0.4154859 0.4085899 0.4056207
```

Doubling time:

```
pred1[3, 'er'] / pred1[2, 'er'] - 1
```

```
## [1] 0.0544534
```

Halving time:

```
1 - pred1[1, 'er'] / pred1[2, 'er']
```

```
## [1] 0.09721166
```

Apparent mitigation effect at reference time:

```
1 - pred0[2, 'er'] / pred1[2, 'er']
```

```
## [1] 0.4085899
```

At later time.

```
1 - pred0[3, 'er'] / pred1[3, 'er']
```

```
## [1] 0.4056207
```

And under higher emission conditions.

```
1 - pred1[2, 'er'] / pred2[2, 'er']
```

```
## [1] 0.349926
```

Later:

```
1 - pred1[3, 'er'] / pred2[3, 'er']
```

```
## [1] 0.3447641
```

Single-pool model

```
p0 <- c(int.f0 = 100, int.r1 = -1.5 - 0.3, int.r2 = -100, int.r3 = -100, int.r5 = -100)
p1 <- c(int.f0 = 100, int.r1 = -1.5      , int.r2 = -100, int.r3 = -100, int.r5 = -100)
p2 <- c(int.f0 = 100, int.r1 = -1.5 + 0.3, int.r2 = -100, int.r3 = -100, int.r5 = -100)
```

```r
pred0 <- alfam2(dat, pars = p0)
```

```
## User-supplied parameters are being used.

## Warning in prepDat(dat, warn = warn): Argument prep.dum = TRUE but there are no variables to convert to dummy variables!
##   Ignoring prep.dum = TRUE.
```

```r
pred1 <- alfam2(dat, pars = p1)
```

```
## User-supplied parameters are being used.

## Warning in prepDat(dat, warn = warn): Argument prep.dum = TRUE but there are no variables to convert to dummy variables!
##   Ignoring prep.dum = TRUE.
```

```r
pred2 <- alfam2(dat, pars = p2)
```

```
## User-supplied parameters are being used.

## Warning in prepDat(dat, warn = warn): Argument prep.dum = TRUE but there are no variables to convert to dummy variables!
##   Ignoring prep.dum = TRUE.
```

Doubling pars effect:

```r
(pred2 / pred1)[, 'er'] - 1
```

```
## [1] 0.218729131 0.048001749 0.002307385
```

Halving:

```r
1 - (pred0 / pred1)[, 'er']
```

```
## [1] 0.31779550 0.17894365 0.04548927
```

Doubling time:

```r
pred1[3, 'er'] / pred1[2, 'er'] - 1
```

```
## [1] 0.04803686
```

Halving time:

```r
1 - pred1[1, 'er'] / pred1[2, 'er']
```

```
## [1] 0.179772
```

Apparent mitigation effect at reference time:

```r
1 - pred0[2, 'er'] / pred1[2, 'er']
```

```
## [1] 0.1789436
```

At later time.

```r
1 - pred0[3, 'er'] / pred1[3, 'er']
```

```
## [1] 0.04548927
```

And under higher emission conditions.

```r
1 - pred1[2, 'er'] / pred2[2, 'er']
```

```
## [1] 0.04580312
```

Later:

```r
1 - pred1[3, 'er'] / pred2[3, 'er']
```

```
## [1] 0.002302073
```

Conclusions

- Predicted emission is much more sensitive to a fixed relative change in emission rate constants than to time
- But mitigation effects drop in response to increases from either time or emission rate constants, although much more for changes in emission rate constants
- For a single-pool first-order model effects of time and r1 are interchageable