ALFAM2 confidence interval calculations

Sasha D. Hafner

2024-02-04

Overview

In January/February 2024 I added some code for calculating confidence intervals to `alfam2()`, given multiple parameter sets that themselves represent the distribution of possible parameter values. This document demonstrates the implementation.

ALFAM2 package

This is from the dev branch. Use line below to remove existing version and install latest dev version.

```
#remove.packages('ALFAM2') ; devtools::install_github('sashahafner/ALFAM2', ref = 'dev')
#detach('package:ALFAM2')
```

```
library(ALFAM2)
packageVersion('ALFAM2')
```

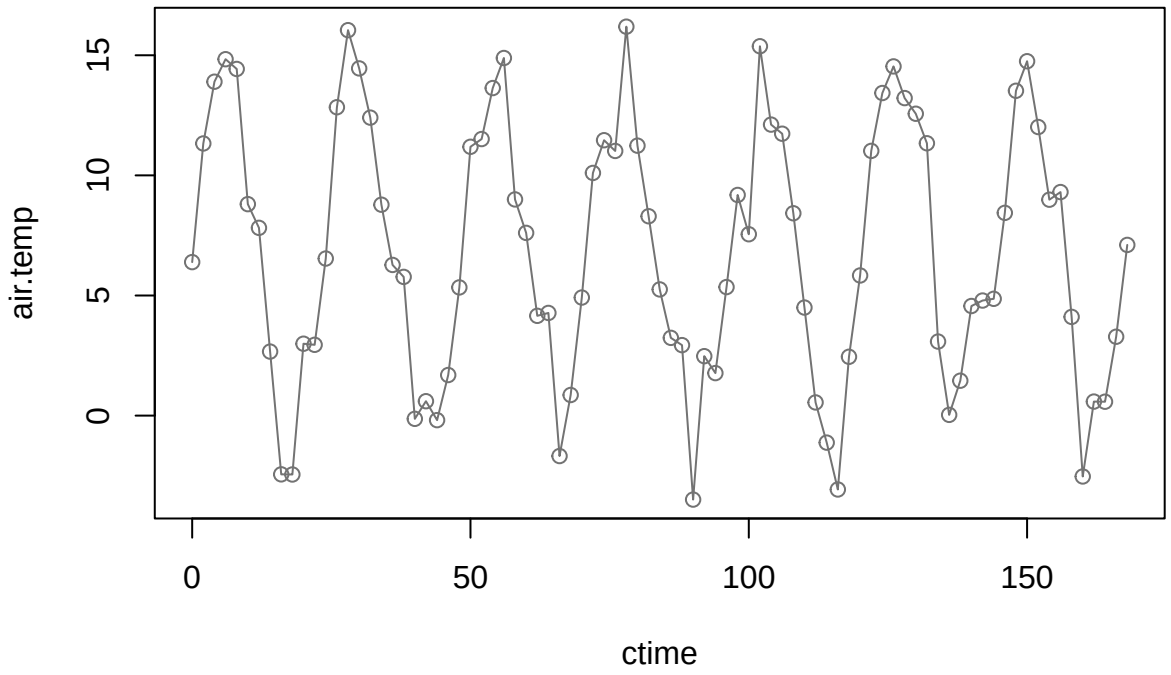
```
## [1] '3.72'
```

I am using v3.70 here.

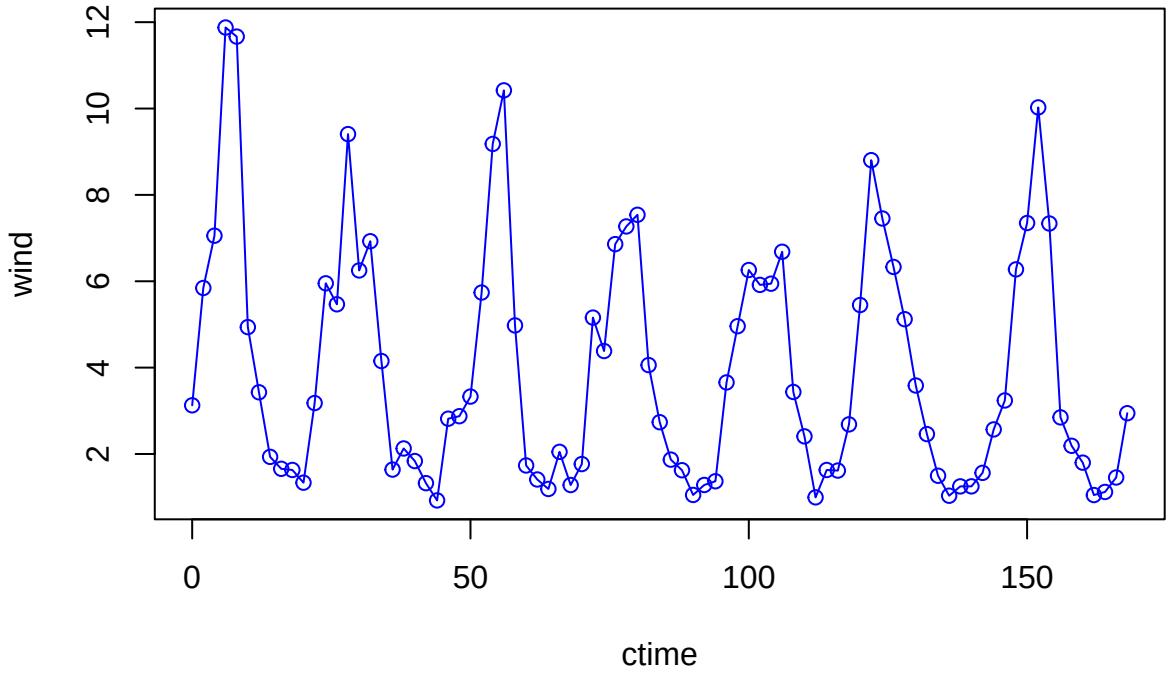
Example 1 with dynamics

```
dat <- data.frame(ctime = 0:84*2, TAN.app = 100, man.dm = 8,
                  air.temp = 7 + 7*sin(0:84*2 * 2*pi/24) + rnorm(85, 0, 2),
                  wind = 10^(0.5 + 0.4*sin(0:84*2 * 2*pi/24) +
                             rnorm(85, 0, 0.12)),
                  app.mthd = "bc")
```

```
plot(air.temp ~ ctime, data = dat, type = 'o', col = 'gray45')
```




```
plot(wind ~ ctime, data = dat, type = 'o', col = 'blue')
```



Normal call without confidence intervals (CI).

```
pred1 <- alfam2(dat, app.name = 'TAN.app', time.name = 'ctime', warn = FALSE)
head(pred1)
```

```
##   app.mthd.ts app.mthd.bc app.mthd.os app.mthd.cs ctime dt         f        s
## 1           0           1           0           0     0  0 54.826376 45.17362
## 2           0           1           0           0     2  2 30.165659 46.28719
## 3           0           1           0           0     4  2 12.191210 46.71676
## 4           0           1           0           0     6  2  4.238835 46.75445
```

```
## 5           0           1           0           0     8  2  1.577815 46.63882
## 6           0           1           0           0    10  2  1.056990 46.47958
##          e      e.int          j        er        f0        r1         r2
## 1  0.00000  0.0000000        NaN 0.0000000 0.5482638 0.1225800 0.01587869
## 2 23.54715 23.5471516 11.7735758 0.2354715 0.5482638 0.2828549 0.01587869
## 3 41.09203 17.5448751  8.7724376 0.4109203 0.5482638 0.4371158 0.01587869
## 4 49.00671  7.9146882  3.9573441 0.4900671 0.5482638 0.5123346 0.01587869
## 5 51.78337  2.7766524  1.3883262 0.5178337 0.5482638 0.4782451 0.01587869
## 6 52.46343  0.6800666  0.3400333 0.5246343 0.5482638 0.1844292 0.01587869
##            r3 f4 r5
## 1 0.002153413  1  0
## 2 0.002153413  1  0
## 3 0.002153413  1  0
## 4 0.002153413  1  0
## 5 0.002153413  1  0
## 6 0.002153413  1  0
```

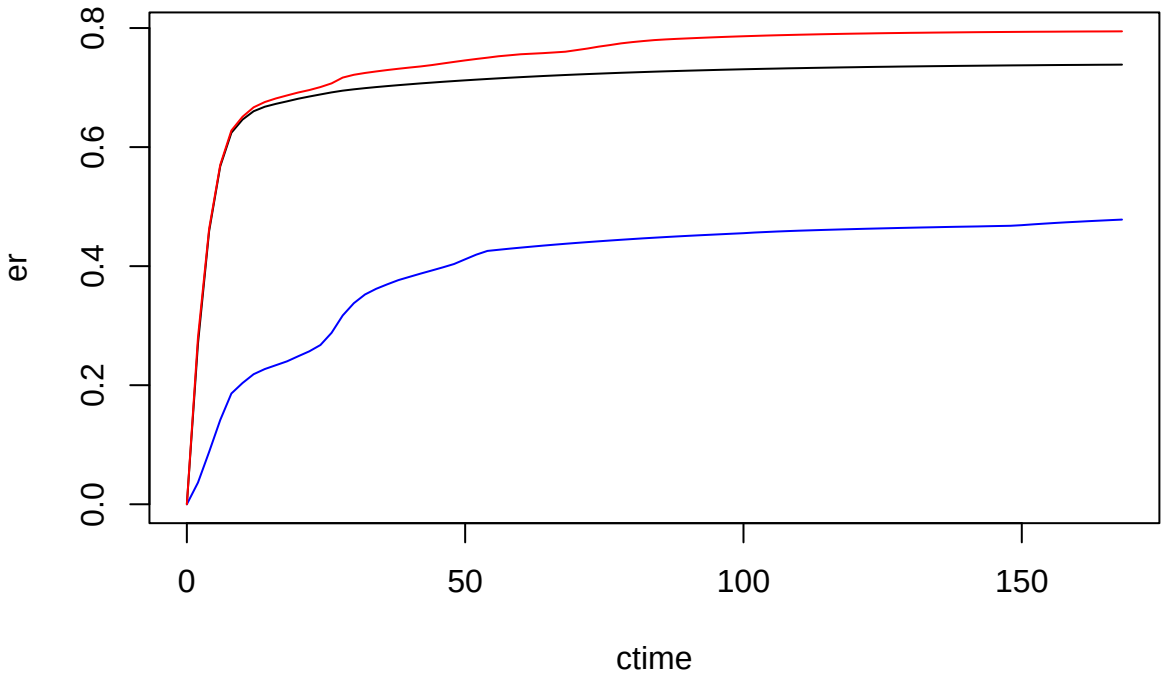
Add CI. They are given in the output with `.lwr` and `.upr` suffixes.

```
predci <- alfam2(dat, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', warn = FALSE, conf.int = 0.90,
                 pars.ci = alfam2pars03var_alpha)
head(predci)
```

```
##    ctime app.mthd.ts app.mthd.bc app.mthd.os app.mthd.cs dt         f        s
## 1      0           0           1           0           0  0 89.331296 10.66870
## 42     2           0           1           0           0  2 52.568534 19.94122
## 53     4           0           1           0           0  2 28.286271 24.62047
## 64     6           0           1           0           0  2 14.659140 26.53051
## 75     8           0           1           0           0  2  7.724482 26.97816
## 2     10           0           1           0           0  2  4.882791 26.76094
##                e         e.int         j            er       f0        r1
## 1  -8.075431e-16 -8.075431e-16      -Inf -8.075431e-18 0.893313 0.1298212
## 42  2.699028e+01  2.699028e+01 13.495140  2.699028e-01 0.893313 0.1935816
## 53  4.587742e+01  1.888714e+01  9.443571  4.587742e-01 0.893313 0.2383350
## 64  5.677851e+01  1.090109e+01  5.450543  5.677851e-01 0.893313 0.2571208
## 75  6.241480e+01  5.636295e+00  2.818148  6.241480e-01 0.893313 0.2487991
## 2   6.462096e+01  2.206162e+00  1.103081  6.462096e-01 0.893313 0.1578034
##            r2          r3 f4         r5        er.lwr       er.upr
## 1  0.07153553 0.004662259  1 0.01584893 -1.799140e-17 2.151580e-17
## 42 0.07153553 0.004662259  1 0.01584893  3.646139e-02 2.802086e-01
## 53 0.07153553 0.004662259  1 0.01584893  8.793869e-02 4.626577e-01
## 64 0.07153553 0.004662259  1 0.01584893  1.415311e-01 5.702058e-01
## 75 0.07153553 0.004662259  1 0.01584893  1.860694e-01 6.278738e-01
## 2  0.07153553 0.004662259  1 0.01584893  2.036588e-01 6.509548e-01
```

By default CI are only returned for variable `er` = relative cumulative emission.

```
plot(er ~ ctime, data = predci, type = 'l', ylim = c(0, max(predci$er.upr)))
lines(er.lwr ~ ctime, data = predci, type = 'l', col = 'blue')
lines(er.upr ~ ctime, data = predci, type = 'l', col = 'red')
```



This 90% CI is quite wide, it is seems a bit strange that the prediction with default parmeters is close to the upper limit, especially at the start. But these are draft parameter values, so this second issue may not be present in the final values. The CIs will likely remain wide. Anyway, the focus of this demo is function behavior and not the values of the CI per se.

We can add any output variables for CI calculation, but `quantile` is applied by variable, so the limits returned from different variables will be from different parameter sets and so should be considered separately. (This is a concept that clearly needs some more development. At the moment I cannot think of how users might even use values for different variables and whether this issue is a problem. But it should not be too difficult to get the function to return values from the same sets instead.) Use the `var.ci` argument for this. Here we request 3 variables.

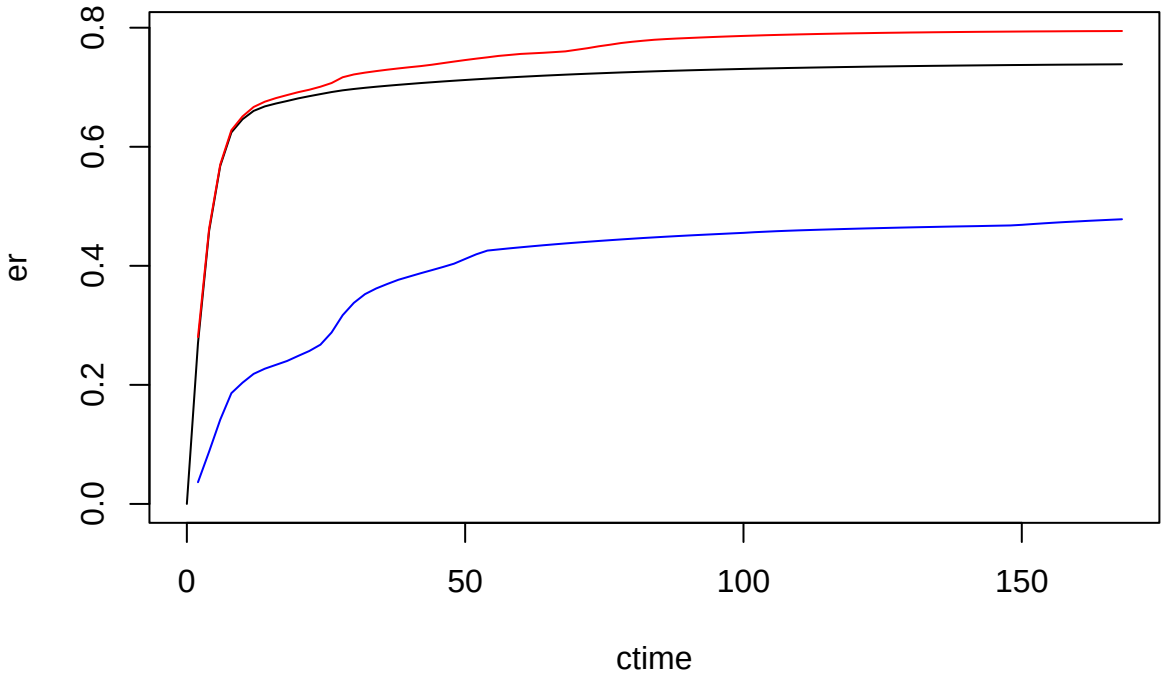
```
predci <- alfam2(dat, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', warn = FALSE, conf.int = 0.90,
                 pars.ci = alfam2pars03var_alpha, var.ci = c('er', 'j', 'r1'))
head(predci)
```

```
##   ctime app.mthd.ts app.mthd.bc app.mthd.os app.mthd.cs dt         f        s
## 1     0           0           1           0           0  0 89.331296 10.66870
## 2     2           0           1           0           0  2 52.568534 19.94122
## 3     4           0           1           0           0  2 28.286271 24.62047
## 4     6           0           1           0           0  2 14.659140 26.53051
## 5     8           0           1           0           0  2  7.724482 26.97816
## 6    10           0           1           0           0  2  4.882791 26.76094
##                e         e.int         j            er       f0        r1
## 1 -8.075431e-16 -8.075431e-16      -Inf -8.075431e-18 0.893313 0.1298212
## 2  2.699028e+01  2.699028e+01 13.495140  2.699028e-01 0.893313 0.1935816
## 3  4.587742e+01  1.888714e+01  9.443571  4.587742e-01 0.893313 0.2383350
## 4  5.677851e+01  1.090109e+01  5.450543  5.677851e-01 0.893313 0.2571208
## 5  6.241480e+01  5.636295e+00  2.818148  6.241480e-01 0.893313 0.2487991
## 6  6.462096e+01  2.206162e+00  1.103081  6.462096e-01 0.893313 0.1578034
##           r2          r3 f4         r5     er.lwr     j.lwr     r1.lwr
## 1 0.07153553 0.004662259  1 0.01584893         NA        NA         NA
## 2 0.07153553 0.004662259  1 0.01584893 0.03646139 1.8230693 0.02394579
```

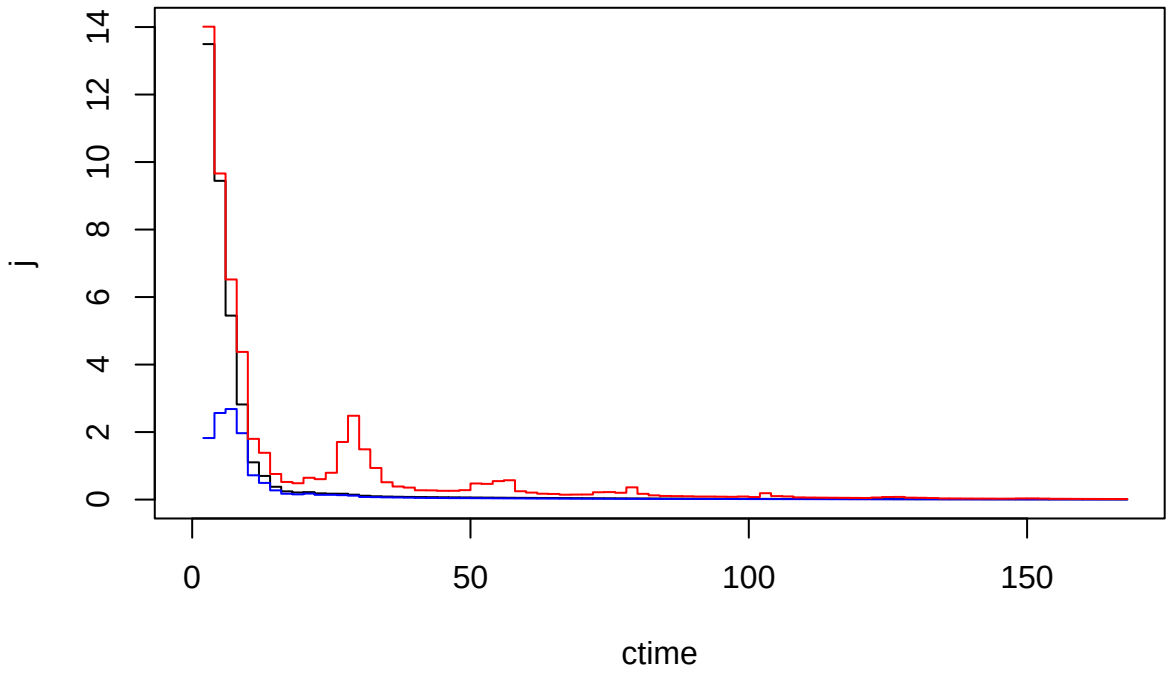
```
## 3 0.07153553 0.004662259  1 0.01584893 0.08793869 2.5656187 0.03394136
## 4 0.07153553 0.004662259  1 0.01584893 0.14153107 2.6818143 0.04024019
## 5 0.07153553 0.004662259  1 0.01584893 0.18606938 1.9670083 0.03737554
## 6 0.07153553 0.004662259  1 0.01584893 0.20365877 0.7195175 0.01384893
##      er.upr     j.upr    r1.upr
## 1        NA        NA        NA
## 2 0.2802086 14.010431 0.2137312
## 3 0.4626577  9.661560 0.2644441
## 4 0.5702058  6.521131 0.2839784
## 5 0.6278738  4.375323 0.2753348
## 6 0.6509548  1.798929 0.1806447
```

Note that times with any `NaN` etc. in one of `var.ci` columns will be dropped before applying the `quantile()` function. So here all `lwr` and `upr` limits are `NA` for time = 0 h.

```
plot(er ~ ctime, data = predci, type = 'l', ylim = c(0, max(na.omit(predci$er.upr))))
lines(er.lwr ~ ctime, data = predci, type = 'l', col = 'blue')
lines(er.upr ~ ctime, data = predci, type = 'l', col = 'red')
```




```
plot(j ~ ctime, data = predci, type = 's', ylim = c(0, max(na.omit(predci$j.upr))))
lines(j.lwr ~ ctime, data = predci, type = 's', col = 'blue')
lines(j.upr ~ ctime, data = predci, type = 's', col = 'red')
```



Example 2 application to multiple groups

Here is a test where each group has a different incorporation time (based on one in vignette).

```
datm <- data.frame(scenario = 1:6, ctime = 168, TAN.app = 50,
                   man.dm = 8, air.temp = 20, wind.2m = 4,
                   app.mthd = 'bc',
                   incorp = 'deep',
                   t.incorp = c(0.1, 1, 6, 24, 168, NA))
```

```
predci <- alfam2(datm, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', time.incorp = 't.incorp', group = 'scenario',
                 conf.int = 0.90,
                 pars.ci = alfam2pars03var_alpha, var.ci = c('er'))
```

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

## Incorporation skipped where it occurred after all intervals, for groups: 5.

## Incorporation applied for groups: 1, 2, 3, 4.

## Warning in alfam2(dat = dat, pars = pars, add.pars = add.pars, app.name = app.name, : Running with 18
## These secondary parameters have been dropped:
##   app.rate.ni.f0
##   man.source.pig.f0
##   man.ph.r1
##   rain.rate.r2
##   man.ph.r3
##   rain.rate.r5
##   wind.sqrt.r1
```

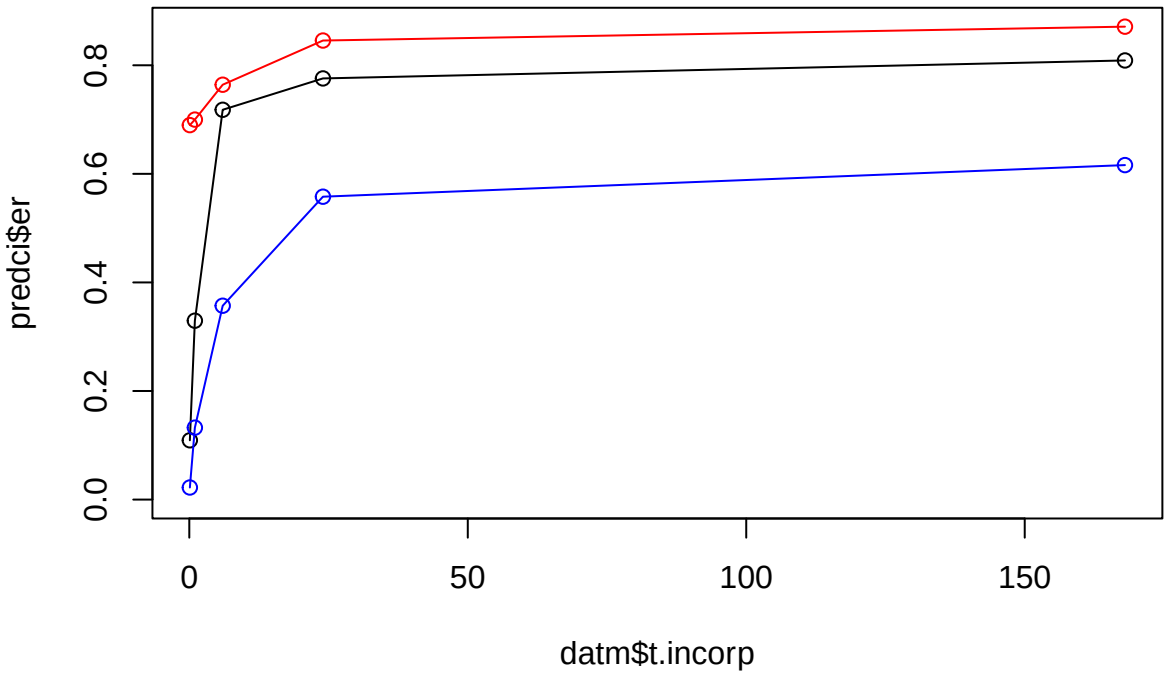
```
predci
```

```
##   scenario ctime app.mthd.ts app.mthd.bc app.mthd.os app.mthd.cs incorp.shallow
## 1        1   168           0           1           0           0              0
```

```
## 2        2   168           0           1           0           0             0
## 3        3   168           0           1           0           0             0
## 4        4   168           0           1           0           0             0
## 5        5   168           0           1           0           0             0
## 6        6   168           0           1           0           0             0
##   incorp.deep  dt            f         s         e     e.int          j
## 1           1 168 9.047443e-34 3.1123681  5.459711  5.459711 0.03249828
## 2           1 168 9.047443e-34 2.3691465 16.474979 16.474979 0.09806535
## 3           1 168 9.047443e-34 1.0132177 35.899590 35.899590 0.21368803
## 4           1 168 9.047443e-34 0.7839046 38.788686 38.788686 0.23088504
## 5           1 168 8.708448e-33 0.4007291 40.443414 40.443414 0.24073461
## 6           1 168 8.708448e-33 0.4007291 40.443414 40.443414 0.24073461
##          er       f0        r1         r2           r3        f4         r5
## 1 0.1091942 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 2 0.3294996 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 3 0.7179918 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 4 0.7757737 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 5 0.8088683 0.893313 0.3904896 0.07153553 4.662259e-03 1.0000000 0.01584893
## 6 0.8088683 0.893313 0.3904896 0.07153553 4.662259e-03 1.0000000 0.01584893
##       er.lwr    er.upr
## 1 0.02222003 0.6897585
## 2 0.13248098 0.6999392
## 3 0.35707203 0.7641794
## 4 0.55788859 0.8455538
## 5 0.61613940 0.8710960
## 6 0.61613940 0.8710960
```

Plot emission versus time of incorporation.

```
plot(datm$t.incorp, predci$er, type = 'o', ylim = c(0, max(predci$er.upr)))
lines(datm$t.incorp, predci$er.lwr, type = 'o', col = 'blue')
lines(datm$t.incorp, predci$er.upr, type = 'o', col = 'red')
```



Notice how the CI is larger for rapid incorporation, because of uncertainty in incorporation parameters that

has a smaller effect when incorporation is done later.

By default the model is run with all the different parameter sets provided in `pars.ci`, which is 100 in this draft object.

```
dim(alfam2pars03var_alpha)
```

```
## [1] 100  25
```

For speed some users might want to sometimes reduce that.

```
predci <- alfam2(datm, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', time.incorp = 't.incorp', group = 'scenario',
                 conf.int = 0.90, pars.ci = alfam2pars03var_alpha,
                 n.ci = 10)
```

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

## Incorporation skipped where it occurred after all intervals, for groups: 5.

## Incorporation applied for groups: 1, 2, 3, 4.

## Warning in alfam2(dat = dat, pars = pars, add.pars = add.pars, app.name = app.name, : Running with 18
## These secondary parameters have been dropped:
##   app.rate.ni.f0
##   man.source.pig.f0
##   man.ph.r1
##   rain.rate.r2
##   man.ph.r3
##   rain.rate.r5
##   wind.sqrt.r1
```

```
predci
```

```
##   scenario ctime app.mthd.ts app.mthd.bc app.mthd.os app.mthd.cs incorp.shallow
## 1        1   168           0           1           0           0              0
## 2        2   168           0           1           0           0              0
## 3        3   168           0           1           0           0              0
## 4        4   168           0           1           0           0              0
## 5        5   168           0           1           0           0              0
## 6        6   168           0           1           0           0              0
##   incorp.deep  dt            f         s         e     e.int          j
## 1           1 168 9.047443e-34 3.1123681  5.459711  5.459711 0.03249828
## 2           1 168 9.047443e-34 2.3691465 16.474979 16.474979 0.09806535
## 3           1 168 9.047443e-34 1.0132177 35.899590 35.899590 0.21368803
## 4           1 168 9.047443e-34 0.7839046 38.788686 38.788686 0.23088504
## 5           1 168 8.708448e-33 0.4007291 40.443414 40.443414 0.24073461
## 6           1 168 8.708448e-33 0.4007291 40.443414 40.443414 0.24073461
##          er       f0        r1         r2           r3        f4         r5
## 1 0.1091942 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 2 0.3294996 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 3 0.7179918 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 4 0.7757737 0.893313 0.3904896 0.07153553 2.989078e-06 0.1038927 0.01584893
## 5 0.8088683 0.893313 0.3904896 0.07153553 4.662259e-03 1.0000000 0.01584893
## 6 0.8088683 0.893313 0.3904896 0.07153553 4.662259e-03 1.0000000 0.01584893
##       er.lwr    er.upr
## 1 0.05736533 0.6322769
## 2 0.10081919 0.6632179
## 3 0.25072978 0.7492937
```

```
## 4 0.42000942 0.8352185
## 5 0.60823885 0.8733644
## 6 0.60823885 0.8733644
```

Example 3 on getting all predictions

When `conf.int=` some number it is used in the `quantile` function. To get all results, use `conf.int = 'all'`. This could be useful to combine uncertainty in parameter values with uncertainty in inputs. Of course then the CIs would have to be constructed outside the `alfam2` function, but it cannot do everything, so I think this is OK.

```
datvar <- data.frame(ctime = 168, TAN.app = 50, man.dm = rnorm(7, mean = 8, sd = 1),
                     air.temp = 20, wind.2m = 3)
```

Here is what we get with numeric `conf.int`.

```
predci <- alfam2(datvar, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', group = 'man.dm',
                 conf.int = 0.90, pars.ci = alfam2pars03var_alpha)
```

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

```
## Warning in prepDat(dat, value = "dummy", warn = warn): Argument prep.dum = TRUE but there are no vari
##   Ignoring prep.dum = TRUE.
```

```
## Warning in alfam2(dat = dat, pars = pars, add.pars = add.pars, app.name = app.name, : Running with 8
## These secondary parameters have been dropped:
##   app.mthd.os.f0
##   app.rate.ni.f0
##   man.source.pig.f0
##   app.mthd.cs.f0
##   app.mthd.bc.r1
##   app.mthd.ts.r1
##   man.ph.r1
##   rain.rate.r2
##   app.mthd.bc.r3
##   app.mthd.cs.r3
##   man.ph.r3
##   incorp.shallow.f4
##   incorp.shallow.r3
##   incorp.deep.f4
##   incorp.deep.r3
##   rain.rate.r5
##   wind.sqrt.r1
```

```
predci
```

```
##     man.dm ctime  dt            f        s        e    e.int         j
## 1 1.073782   168 168 7.899604e-11 1.443423 26.08294 26.08294 0.1552556
## 6 2.911664   168 168 1.240130e-09 1.468511 26.43394 26.43394 0.1573449
## 4 2.338558   168 168 5.602136e-10 1.448462 26.54577 26.54577 0.1580105
## 5 2.416732   168 168 6.263423e-10 1.450651 26.54040 26.54040 0.1579786
## 2 1.190478   168 168 9.587530e-11 1.441148 26.17511 26.17511 0.1558042
## 3 1.622635   168 168 1.918083e-10 1.437924 26.42267 26.42267 0.1572778
## 7 3.577800   168 168 2.928975e-09 1.501681 26.12535 26.12535 0.1555080
##          er        f0         r1         r2          r3 f4         r5    er.lwr
## 1 0.5216588 0.8360737 0.08914666 0.07153553 0.002038241  1 0.01584893 0.4172827
```

```
## 6 0.5286787 0.9316926 0.07340091 0.07153553 0.002038241  1 0.01584893 0.4279732
## 4 0.5309153 0.9093927 0.07798678 0.07153553 0.002038241  1 0.01584893 0.4320610
## 5 0.5308080 0.9127818 0.07734476 0.07153553 0.002038241  1 0.01584893 0.4315951
## 2 0.5235021 0.8444553 0.08805336 0.07153553 0.002038241  1 0.01584893 0.4211888
## 3 0.5284534 0.8724794 0.08412002 0.07153553 0.002038241  1 0.01584893 0.4328136
## 7 0.5225069 0.9511779 0.06840842 0.07153553 0.002038241  1 0.01584893 0.4217404
##      er.upr
## 1 0.7266954
## 6 0.7999278
## 4 0.7758685
## 5 0.7789009
## 2 0.7348973
## 3 0.7463088
## 7 0.8138980
```

Not so useful.

The alternative:

```
predci <- alfam2(datvar, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', group = 'man.dm',
                 conf.int = 'all', pars.ci = alfam2pars03var_alpha, n.ci = 3)
```

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

```
## Warning in prepDat(dat, value = "dummy", warn = warn): Argument prep.dum = TRUE but there are no vari
##   Ignoring prep.dum = TRUE.
```

```
## Warning in alfam2(dat = dat, pars = pars, add.pars = add.pars, app.name = app.name, : Running with 8
## These secondary parameters have been dropped:
##   app.mthd.os.f0
##   app.rate.ni.f0
##   man.source.pig.f0
##   app.mthd.cs.f0
##   app.mthd.bc.r1
##   app.mthd.ts.r1
##   man.ph.r1
##   rain.rate.r2
##   app.mthd.bc.r3
##   app.mthd.cs.r3
##   man.ph.r3
##   incorp.shallow.f4
##   incorp.shallow.r3
##   incorp.deep.f4
##   incorp.deep.r3
##   rain.rate.r5
##   wind.sqrt.r1
```

```
predci
```

```
##      man.dm ctime  dt            f        s        e    e.int         j
## 1  1.073782   168 168 3.449635e-09 1.688835 24.00211 24.00211 0.1428697
## 2  2.911664   168 168 2.967011e-09 1.617144 25.35212 25.35212 0.1509055
## 3  2.338558   168 168 3.123402e-09 1.636014 25.00299 25.00299 0.1488273
## 4  2.416732   168 168 3.102220e-09 1.633275 25.05399 25.05399 0.1491309
## 5  1.190478   168 168 3.421439e-09 1.683218 24.10981 24.10981 0.1435108
## 6  1.622635   168 168 3.312983e-09 1.663807 24.47992 24.47992 0.1457138
```

```
## 7  3.577800   168 168 2.784367e-09 1.598344 25.69361 25.69361 0.1529382
## 8  1.073782   168 168 1.369601e-13 1.221507 29.69438 29.69438 0.1767523
## 9  2.911664   168 168 1.106495e-11 1.288094 29.17399 29.17399 0.1736547
## 10 2.338558   168 168 3.157313e-12 1.254453 29.56001 29.56001 0.1759525
## 11 2.416732   168 168 3.767910e-12 1.258495 29.51676 29.51676 0.1756950
## 12 1.190478   168 168 1.872657e-13 1.221482 29.73563 29.73563 0.1769978
## 13 1.622635   168 168 5.714628e-13 1.227360 29.78367 29.78367 0.1772838
## 14 3.577800   168 168 4.234191e-11 1.336788 28.56206 28.56206 0.1700123
## 15 1.073782   168 168 6.341499e-06 1.443548 26.52070 26.52070 0.1578613
## 16 2.911664   168 168 1.108759e-05 1.363335 28.95752 28.95752 0.1723662
## 17 2.338558   168 168 9.546266e-06 1.368982 28.65415 28.65415 0.1705604
## 18 2.416732   168 168 9.751645e-06 1.367521 28.71193 28.71193 0.1709043
## 19 1.190478   168 168 6.631586e-06 1.431391 26.84109 26.84109 0.1597684
## 20 1.622635   168 168 7.713014e-06 1.397052 27.77829 27.77829 0.1653470
## 21 3.577800   168 168 1.301387e-05 1.367313 29.05949 29.05949 0.1729732
##           er        f0         r1         r2          r3 f4         r5
## 1  0.4800423 0.8932445 0.06671464 0.07188135 0.001632210  1 0.01584893
## 2  0.5070425 0.9457887 0.06795197 0.07188135 0.001632210  1 0.01584893
## 3  0.5000598 0.9327671 0.06756369 0.07188135 0.001632210  1 0.01584893
## 4  0.5010798 0.9347009 0.06761652 0.07188135 0.001632210  1 0.01584893
## 5  0.4821961 0.8976126 0.06679253 0.07188135 0.001632210  1 0.01584893
## 6  0.4895983 0.9124368 0.06708177 0.07188135 0.001632210  1 0.01584893
## 7  0.5138723 0.9579308 0.06840608 0.07188135 0.001632210  1 0.01584893
## 8  0.5938876 0.8901392 0.12211827 0.07677895 0.001966689  1 0.01584893
## 9  0.5834799 0.9609496 0.09643193 0.07677895 0.001966689  1 0.01584893
## 10 0.5912003 0.9456617 0.10380112 0.07677895 0.001966689  1 0.01584893
## 11 0.5903351 0.9480393 0.10276369 0.07677895 0.001966689  1 0.01584893
## 12 0.5947127 0.8968496 0.12030083 0.07677895 0.001966689  1 0.01584893
## 13 0.5956735 0.9186330 0.11380278 0.07677895 0.001966689  1 0.01584893
## 14 0.5712413 0.9735506 0.08852141 0.07677895 0.001966689  1 0.01584893
## 15 0.5304140 0.8470577 0.05073061 0.04280767 0.002553698  1 0.01584893
## 16 0.5791505 0.9734582 0.04823284 0.04280767 0.002553698  1 0.01584893
## 17 0.5730830 0.9531420 0.04899823 0.04280767 0.002553698  1 0.01584893
## 18 0.5742385 0.9566052 0.04889312 0.04280767 0.002553698  1 0.01584893
## 19 0.5368219 0.8619688 0.05056824 0.04280767 0.002553698  1 0.01584893
## 20 0.5555659 0.9068910 0.04997144 0.04280767 0.002553698  1 0.01584893
## 21 0.5811899 0.9864445 0.04735822 0.04280767 0.002553698  1 0.01584893
```

21 rows here, deliberately small so we can look at the results.

More plausible usage would have at least 100 of each I suppose, for 10000 rows in the output.

```
datvar <- data.frame(ctime = 168, TAN.app = 50, man.dm = rnorm(100, mean = 8, sd = 1),
                     air.temp = 20, wind.2m = 3)
```

```
predci <- alfam2(datvar, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', group = 'man.dm',
                 conf.int = 'all', pars.ci = alfam2pars03var_alpha, n.ci = 100)
```

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

```
## Warning in prepDat(dat, value = "dummy", warn = warn): Argument prep.dum = TRUE but there are no vari
##   Ignoring prep.dum = TRUE.
```

```
## Warning in alfam2(dat = dat, pars = pars, add.pars = add.pars, app.name = app.name, : Running with 8
## These secondary parameters have been dropped:
##   app.mthd.os.f0
```

```
##   app.rate.ni.f0
##   man.source.pig.f0
##   app.mthd.cs.f0
##   app.mthd.bc.r1
##   app.mthd.ts.r1
##   man.ph.r1
##   rain.rate.r2
##   app.mthd.bc.r3
##   app.mthd.cs.r3
##   man.ph.r3
##   incorp.shallow.f4
##   incorp.shallow.r3
##   incorp.deep.f4
##   incorp.deep.r3
##   rain.rate.r5
##   wind.sqrt.r1
```

```
head(predci)
```

```
##      man.dm ctime  dt            f         s        e    e.int         j
## 1 1.5208958   168 168 7.353898e-07 0.6943602 38.24583 38.24583 0.2276537
## 2 0.3975235   168 168 1.000592e-09 0.8086543 35.34433 35.34433 0.2103829
## 3 4.1267618   168 168 3.741188e-03 1.1164222 35.43492 35.43492 0.2109221
## 4 3.2826757   168 168 4.955516e-04 0.8858345 37.32657 37.32657 0.2221820
## 5 0.5701461   168 168 3.251229e-09 0.7777425 36.01161 36.01161 0.2143548
## 6 0.0319183   168 168 6.546678e-11 0.8898239 33.66323 33.66323 0.2003764
##          er        f0         r1         r2          r3 f4         r5
## 1 0.7649165 0.8798442 0.08853607 0.01805249 0.002442449  1 0.01584893
## 2 0.7068867 0.7586679 0.12693862 0.01805249 0.002442449  1 0.01584893
## 3 0.7086984 0.9811534 0.03838411 0.01805249 0.002442449  1 0.01584893
## 4 0.7465314 0.9650092 0.05031801 0.01805249 0.002442449  1 0.01584893
## 5 0.7202322 0.7816527 0.12010174 0.01805249 0.002442449  1 0.01584893
## 6 0.6732646 0.7047868 0.14273109 0.01805249 0.002442449  1 0.01584893
```

And then, externally, for a 95% confidence interval that includes uncertainty in both inputs (only DM here) and parameters, we can use `quatile()`:

```
quantile(predci$er, c(0.05, 0.95))
```

```
##        5%       95%
## 0.4078082 0.7696533
```

Error messages

The calls below demonstrate some errors.

```
predci <- alfam2(datvar, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', group = 'man.dm',
                 conf.int = 'all', n.ci = 100)
```

```
## Error: Expect class "data.frame, matrix, array" for argument pars.ci but got "NULL".
```

```
predci <- alfam2(datvar, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', group = 'man.dm',
                 conf.int = 'all', pars.ci = alfam2pars03var_alpha, n.ci = 1000)
```

```
## Error: Expect values within the range "0, 100" for argument n.ci but got "1000, 1000".
```

```r
predci <- alfam2(datvar, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', group = 'man.dm',
                 conf.int = 'all', pars.ci = alfam2pars03var_alpha, n.ci = 10,
                 var.ci = 'blahblah')
```

```
## Error: Expect one of the following values "f0, r1, r2, r3, f4, r5, f, s, j, ei, e, er" for argument
```