ALFAM2 confidence interval calculations

Sasha D. Hafner

2024-02-02

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')
```

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

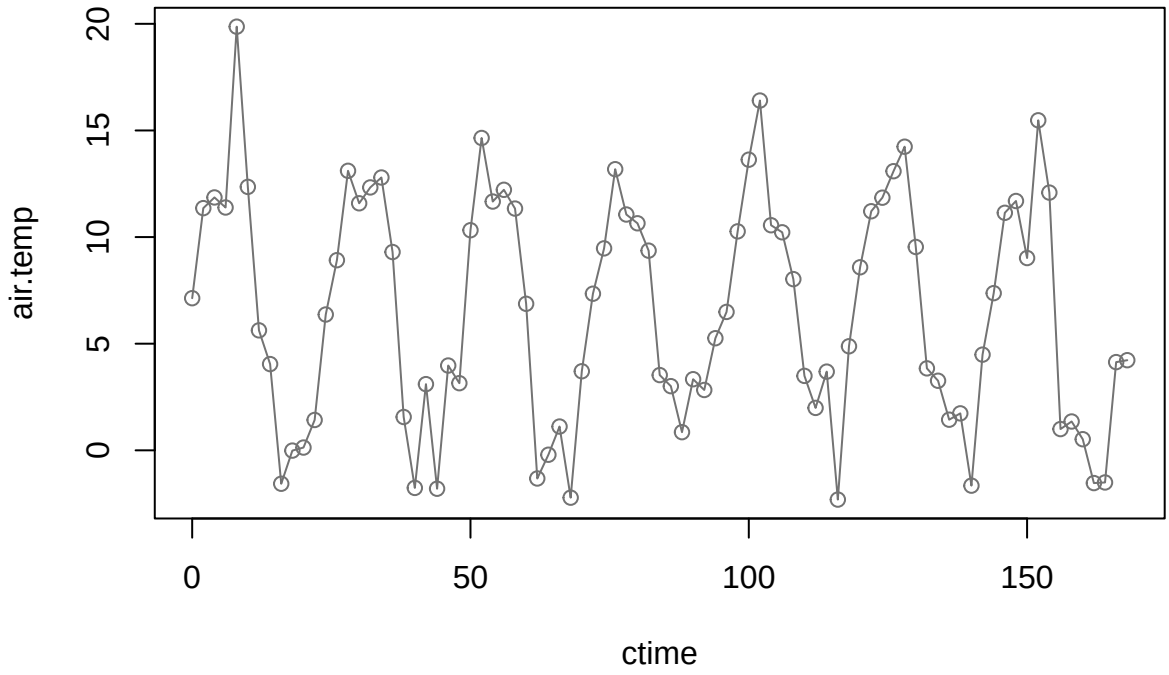
```
## [1] '3.70'
```

I am using v3.70 here.

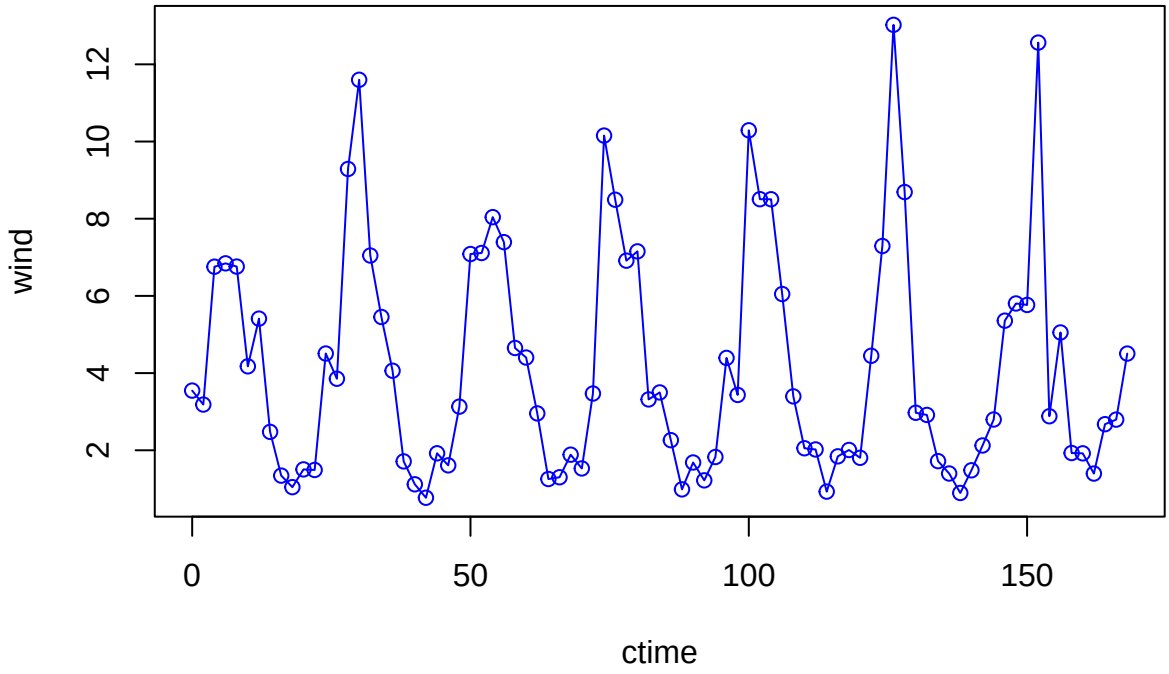
Example 1 with dynamics

```
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 54.8263762 45.17362
## 2           0           1           0           0     2  2 30.0826425 46.28557
## 3           0           1           0           0     4  2 15.6945631 46.78726
## 4           0           1           0           0     6  2  8.5865940 46.95958
```

```
## 5           0           1           0           0     8  2  0.7532811 46.85969
## 6           0           1           0           0    10  2  0.3722686 46.67543
##          e      e.int          j        er        f0        r1         r2
## 1  0.00000  0.0000000        NaN 0.0000000 0.5482638 0.1390482 0.01587869
## 2 23.63179 23.6317920 11.8158960 0.2363179 0.5482638 0.2842328 0.01587869
## 3 37.51818 13.8863867  6.9431934 0.3751818 0.5482638 0.3094383 0.01587869
## 4 44.45383  6.9356495  3.4678248 0.4445383 0.5482638 0.2856774 0.01587869
## 5 52.38703  7.9332013  3.9666007 0.5238703 0.5482638 1.2008808 0.01587869
## 6 52.95230  0.5652689  0.2826344 0.5295230 0.5482638 0.3365327 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
```

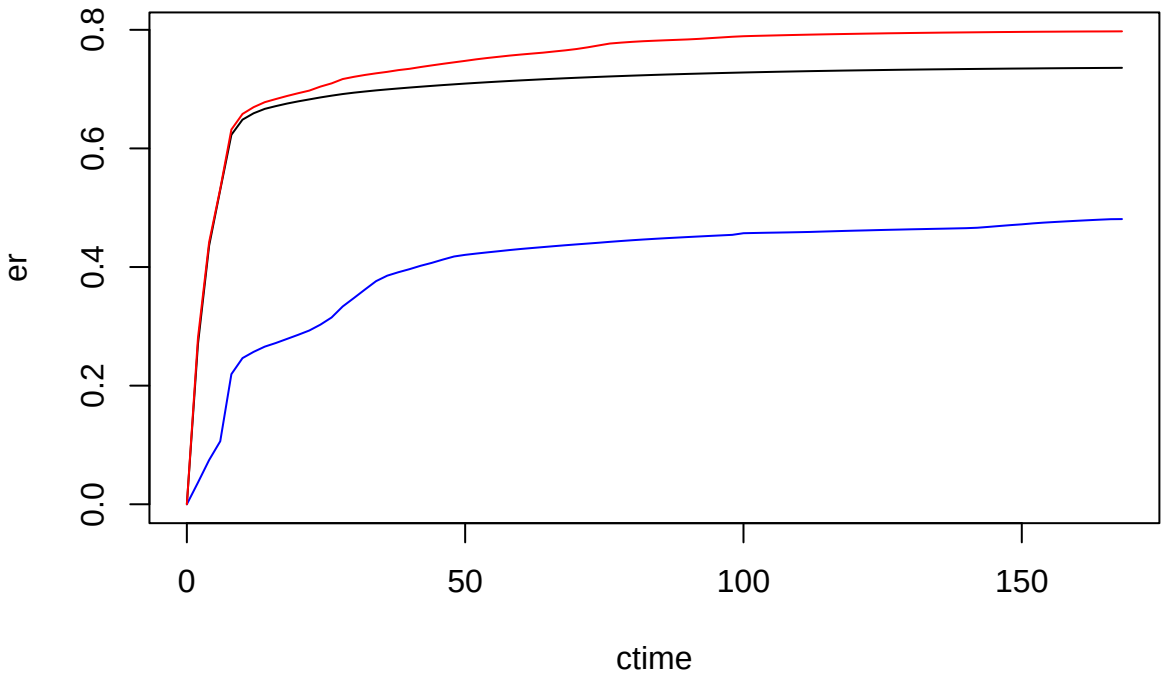
Add CI. They are given in the output with `.lwr` and `.upr` suffixes.

```
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 52.521241 19.93721
## 53     4           0           1           0           0  2 30.386689 24.79541
## 64     6           0           1           0           0  2 17.848676 27.09601
## 75     8           0           1           0           0  2  7.144141 27.64068
## 2     10           0           1           0           0  2  4.065518 27.29376
##                 e         e.int         j            er       f0        r1
## 1  -8.075431e-16 -8.075431e-16      -Inf -8.075431e-18 0.893313 0.1378810
## 42  2.704163e+01  2.704163e+01 13.520815  2.704163e-01 0.893313 0.1940316
## 53  4.360017e+01  1.655854e+01  8.279271  4.360017e-01 0.893313 0.2020710
## 64  5.301020e+01  9.410028e+00  4.705014  5.301020e-01 0.893313 0.1945022
## 75  6.229850e+01  9.288306e+00  4.644153  6.229850e-01 0.893313 0.3862829
## 2   6.485227e+01  2.553765e+00  1.276882  6.485227e-01 0.893313 0.2103401
##            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  3.668558e-02 2.807108e-01
## 53 0.07153553 0.004662259  1 0.01584893  7.457949e-02 4.419254e-01
## 64 0.07153553 0.004662259  1 0.01584893  1.062324e-01 5.319888e-01
## 75 0.07153553 0.004662259  1 0.01584893  2.193669e-01 6.317907e-01
## 2  0.07153553 0.004662259  1 0.01584893  2.464575e-01 6.581034e-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 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.

We can add any output variables for CI calculation, but `quantile` is applied by variable, so the limits may not associate between variables. Use the `var.ci` argument for this. Here we request 3 variables.

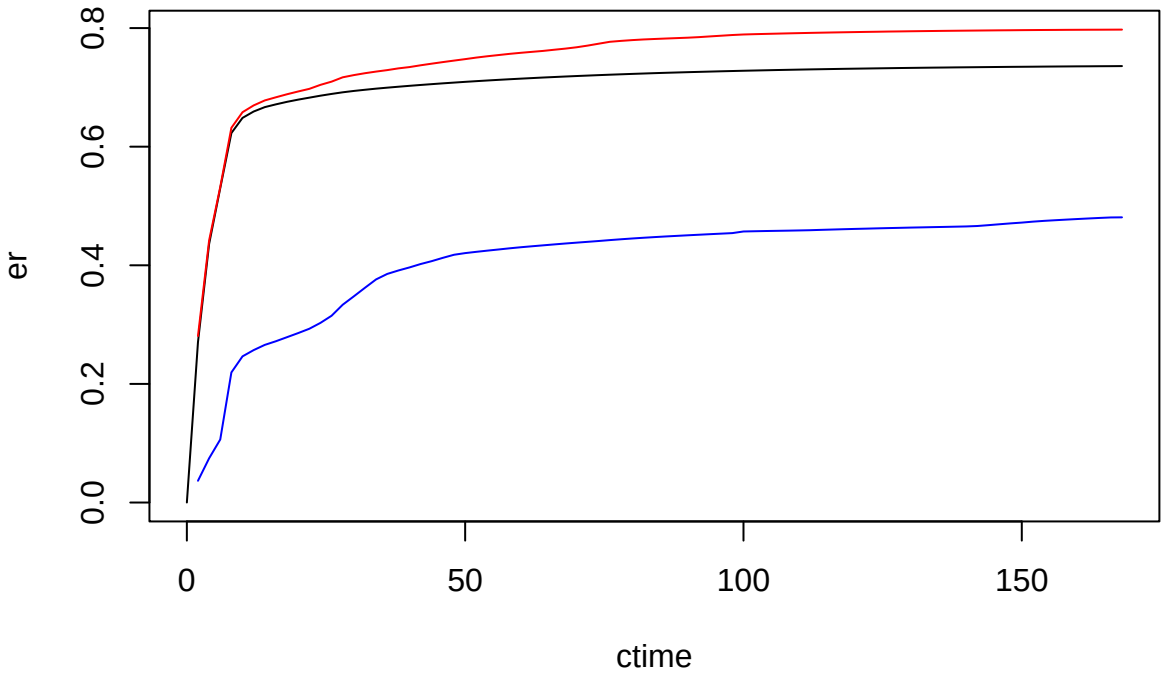
```
predci <- alfam2(dat, pars = alfam2pars03_alpha, app.name = 'TAN.app',
                 time.name = 'ctime', warn = FALSE, conf.int = 0.90,
                 pars.ci = alfam2pars03var_alpha, var.ci = c('er', 'j', 'r1'))
head(predci)
```

```
##   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 52.521241 19.93721
## 3     4           0           1           0           0  2 30.386689 24.79541
## 4     6           0           1           0           0  2 17.848676 27.09601
## 5     8           0           1           0           0  2  7.144141 27.64068
## 6    10           0           1           0           0  2  4.065518 27.29376
##                e         e.int         j            er       f0        r1
## 1 -8.075431e-16 -8.075431e-16      -Inf -8.075431e-18 0.893313 0.1378810
## 2  2.704163e+01  2.704163e+01 13.520815  2.704163e-01 0.893313 0.1940316
## 3  4.360017e+01  1.655854e+01  8.279271  4.360017e-01 0.893313 0.2020710
## 4  5.301020e+01  9.410028e+00  4.705014  5.301020e-01 0.893313 0.1945022
## 5  6.229850e+01  9.288306e+00  4.644153  6.229850e-01 0.893313 0.3862829
## 6  6.485227e+01  2.553765e+00  1.276882  6.485227e-01 0.893313 0.2103401
##           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 0.03668558 1.8342788 0.02402113
## 3 0.07153553 0.004662259  1 0.01584893 0.07457949 1.8682554 0.02537820
## 4 0.07153553 0.004662259  1 0.01584893 0.10623240 1.5797950 0.02409998
## 5 0.07153553 0.004662259  1 0.01584893 0.21936689 3.1096022 0.08384477
## 6 0.07153553 0.004662259  1 0.01584893 0.24645751 0.9329556 0.02679593
##      er.upr     j.upr    r1.upr
```

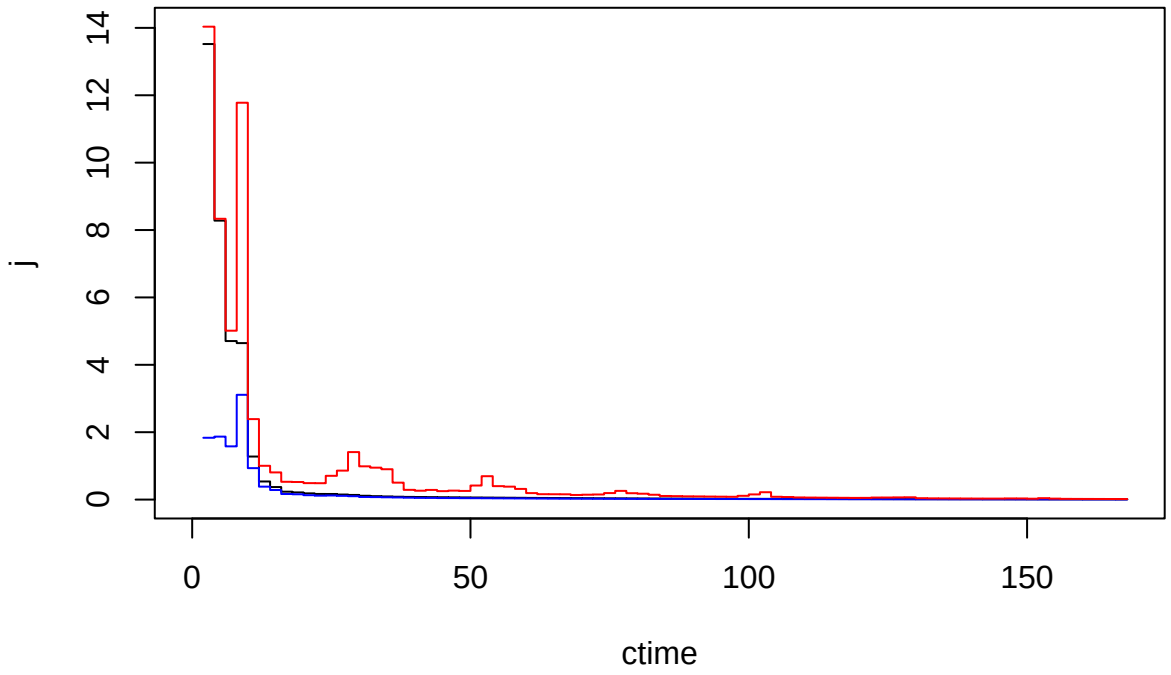
```
## 1        NA        NA        NA
## 2 0.2807108 14.035539 0.2141327
## 3 0.4419254  8.334237 0.2222451
## 4 0.5319888  5.011933 0.2145525
## 5 0.6317907 11.778252 0.5264302
## 6 0.6581034  2.388241 0.2339109
```

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
```

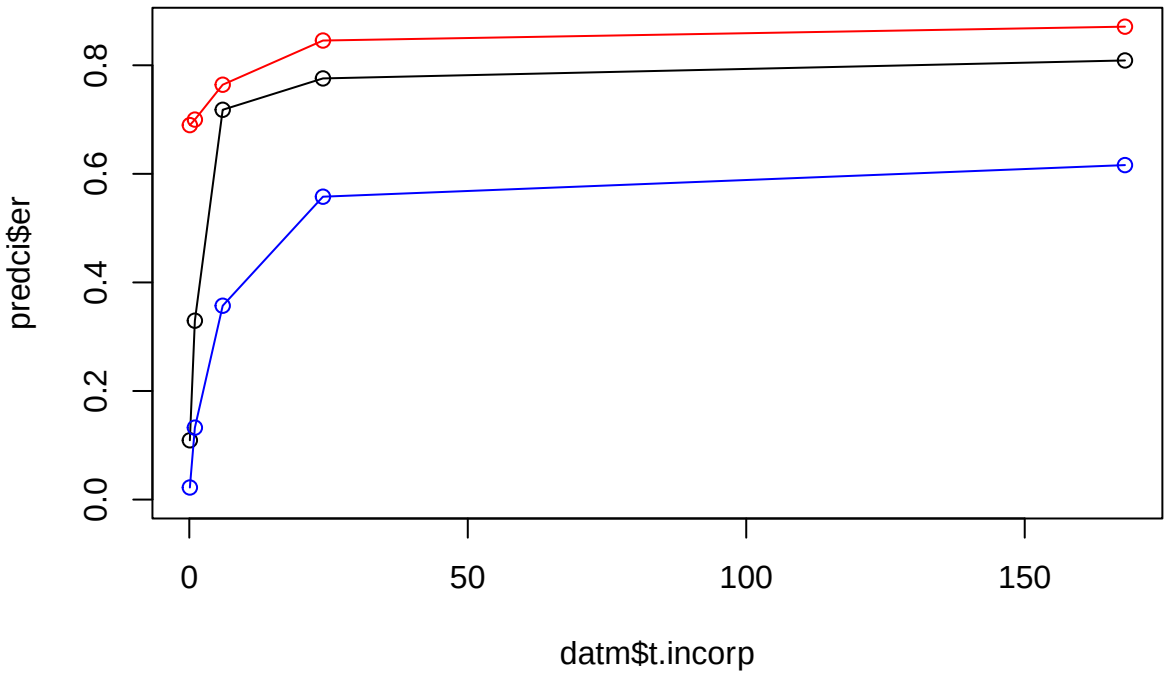
```
predci
```

```
##   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 0.07417709 0.6150101
## 2 0.16421552 0.6488846
## 3 0.40341700 0.7489967
```

```
## 4 0.58057948 0.8375432
## 5 0.65529579 0.8683998
## 6 0.65529579 0.8683998
```

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
## 2 1.5691616   168 168 1.763841e-10 1.437899 26.39969 26.39969 0.1571410
## 1 0.9408855   168 168 6.314481e-11 1.446785 25.96412 25.96412 0.1545483
## 3 2.2014624   168 168 4.594920e-10 1.445081 26.54688 26.54688 0.1580172
## 5 2.8584257   168 168 1.154452e-09 1.466282 26.45097 26.45097 0.1574462
## 4 2.5088351   168 168 7.133991e-10 1.453464 26.52984 26.52984 0.1579157
## 6 2.9758383   168 168 1.351106e-09 1.471288 26.41176 26.41176 0.1572129
## 7 3.1536849   168 168 1.707679e-09 1.479482 26.34134 26.34134 0.1567937
##          er        f0         r1         r2          r3 f4         r5    er.lwr
## 2 0.5279938 0.8692610 0.08459702 0.07153553 0.002038241  1 0.01584893 0.4312394
```

```
## 1 0.5192823 0.8260906 0.09040830 0.07153553 0.002038241  1 0.01584893 0.4126322
## 3 0.5309377 0.9031623 0.07912560 0.07153553 0.002038241  1 0.01584893 0.4327954
## 5 0.5290193 0.9298567 0.07381530 0.07153553 0.002038241  1 0.01584893 0.4284118
## 4 0.5305968 0.9166273 0.07659512 0.07153553 0.002038241  1 0.01584893 0.4310053
## 6 0.5282353 0.9338464 0.07290449 0.07153553 0.002038241  1 0.01584893 0.4274310
## 7 0.5268267 0.9394896 0.07154623 0.07153553 0.002038241  1 0.01584893 0.4258562
##      er.upr
## 2 0.7459607
## 1 0.7140840
## 3 0.7706593
## 5 0.7971671
## 4 0.7823668
## 6 0.8031166
## 7 0.8111737
```

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  1.5691616   168 168 2.633727e-11 1.509179 25.61662 25.61662 0.1524799
## 2  0.9408855   168 168 2.127090e-11 1.533725 25.06202 25.06202 0.1491787
## 3  2.2014624   168 168 3.221958e-11 1.495312 25.95800 25.95800 0.1545119
## 4  2.8584257   168 168 3.931013e-11 1.489140 26.14748 26.14748 0.1556398
## 5  2.5088351   168 168 3.540266e-11 1.491555 26.06411 26.06411 0.1551435
## 6  2.9758383   168 168 4.069571e-11 1.488710 26.16782 26.16782 0.1557609
```

```
## 7  3.1536849   168 168 4.286855e-11 1.488378 26.19220 26.19220 0.1559060
## 8  1.5691616   168 168 3.893080e-12 1.570459 23.79684 23.79684 0.1416479
## 9  0.9408855   168 168 1.089956e-11 1.638066 22.69310 22.69310 0.1350780
## 10 2.2014624   168 168 1.268085e-12 1.502524 24.90072 24.90072 0.1482185
## 11 2.8584257   168 168 3.585765e-13 1.432435 26.03557 26.03557 0.1549737
## 12 2.5088351   168 168 7.113184e-13 1.469646 25.43348 25.43348 0.1513898
## 13 2.9758383   168 168 2.829713e-13 1.419989 26.23677 26.23677 0.1561712
## 14 3.1536849   168 168 1.963798e-13 1.401195 26.54045 26.54045 0.1579789
## 15 1.5691616   168 168 5.877430e-08 1.858661 22.59909 22.59909 0.1345184
## 16 0.9408855   168 168 4.474932e-08 1.873853 22.12197 22.12197 0.1316784
## 17 2.2014624   168 168 7.618880e-08 1.852031 22.91308 22.91308 0.1363874
## 18 2.8584257   168 168 9.842505e-08 1.852649 23.09395 23.09395 0.1374640
## 19 2.5088351   168 168 8.602350e-08 1.851470 23.01424 23.01424 0.1369895
## 20 2.9758383   168 168 1.029004e-07 1.853438 23.11305 23.11305 0.1375776
## 21 3.1536849   168 168 1.099929e-07 1.854980 23.13520 23.13520 0.1377095
##           er        f0         r1         r2          r3 f4         r5
## 1  0.5123323 0.9328858 0.08205276 0.08581978 0.001952389  1 0.01584893
## 2  0.5012403 0.9015576 0.08312113 0.08581978 0.001952389  1 0.01584893
## 3  0.5191600 0.9548599 0.08099140 0.08581978 0.001952389  1 0.01584893
## 4  0.5229496 0.9703467 0.07990319 0.08581978 0.001952389  1 0.01584893
## 5  0.5212823 0.9628852 0.08048043 0.08581978 0.001952389  1 0.01584893
## 6  0.5233565 0.9725098 0.07971026 0.08581978 0.001952389  1 0.01584893
## 7  0.5238440 0.9754972 0.07941889 0.08581978 0.001952389  1 0.01584893
## 8  0.4759369 0.8509863 0.08644068 0.09226459 0.002068896  1 0.01584893
## 9  0.4538620 0.8320072 0.08017831 0.09226459 0.002068896  1 0.01584893
## 10 0.4980143 0.8682653 0.09323707 0.09226459 0.002068896  1 0.01584893
## 11 0.5207115 0.8843872 0.10086516 0.09226459 0.002068896  1 0.01584893
## 12 0.5086697 0.8760334 0.09673141 0.09226459 0.002068896  1 0.01584893
## 13 0.5247353 0.8870812 0.10229276 0.09226459 0.002068896  1 0.01584893
## 14 0.5308090 0.8910573 0.10449376 0.09226459 0.002068896  1 0.01584893
## 15 0.4519818 0.8864363 0.05595646 0.06571638 0.001374906  1 0.01584893
## 16 0.4424393 0.8526237 0.05734776 0.06571638 0.001374906  1 0.01584893
## 17 0.4582615 0.9134322 0.05459033 0.06571638 0.001374906  1 0.01584893
## 18 0.4618790 0.9352011 0.05320623 0.06571638 0.001374906  1 0.01584893
## 19 0.4602847 0.9243389 0.05393833 0.06571638 0.001374906  1 0.01584893
## 20 0.4622609 0.9385116 0.05296259 0.06571638 0.001374906  1 0.01584893
## 21 0.4627039 0.9432262 0.05259566 0.06571638 0.001374906  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 1.0539678   168 168 2.441839e-12 1.437001 25.73722 25.73722 0.1531977
## 2 2.1889662   168 168 9.061742e-12 1.401410 26.70702 26.70702 0.1589703
## 3 2.6021509   168 168 1.415617e-11 1.398151 26.87784 26.87784 0.1599871
## 4 0.9503519   168 168 2.151613e-12 1.442483 25.60694 25.60694 0.1524223
## 5 2.5470791   168 168 1.335094e-11 1.398329 26.85991 26.85991 0.1598804
## 6 1.8398556   168 168 6.138962e-12 1.407900 26.49231 26.49231 0.1576923
##          er        f0         r1        r2          r3 f4         r5
## 1 0.5147443 0.8194071 0.10051846 0.0807382 0.002094503  1 0.01584893
## 2 0.5341403 0.8888858 0.09319750 0.0807382 0.002094503  1 0.01584893
## 3 0.5375568 0.9076982 0.09066686 0.0807382 0.002094503  1 0.01584893
## 4 0.5121389 0.8116193 0.10121479 0.0807382 0.002094503  1 0.01584893
## 5 0.5371981 0.9053669 0.09100015 0.0807382 0.002094503  1 0.01584893
## 6 0.5298463 0.8704547 0.09539067 0.0807382 0.002094503  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%
## 0.4120524 0.7762258
```

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
```