Orange tree growth: A demonstration and evaluation of nonlinear mixed-effects models in R, ADMB, and BUGS

Arni Magnusson, Mark Maunder, and Ben Bolker

February 11, 2013

Contents

1 Introduction

This document serves two purposes:

- Demonstrate how simple nonlinear mixed-effects models can be fitted in R, AD Model Builder, and BUGS

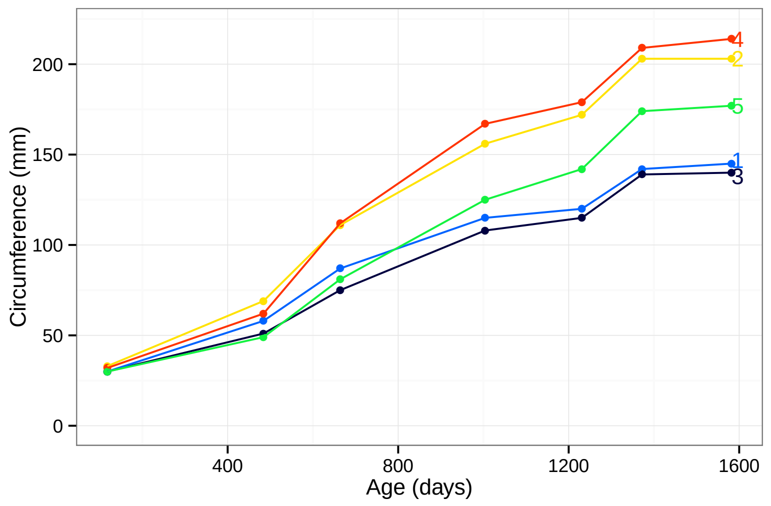


Figure 1. Growth of orange trees. Trunk circumference at breast height of 5 trees measured at 7 different ages.

- Evaluate the estimation performance of models implemented in R and AD Model Builder

2 Data

The data describe the growth of orange trees (Table 1, Figure 1). The trunk circumference of 5 trees is measured at 7 different ages, giving a total of 35 datapoints. These data were used as example data by Pinheiro and Bates (2000, Ch. 8) and have been discussed extensively elsewhere, e.g. Madsen and Thyregod (2010). The `Orange` data object is among the core datasets that come with R.

Table 1. Growth of orange trees. Trunk circumference at breast height of 5 trees measured at 7 different ages.

Age	Circumference (mm)				
(days)	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5
118	30	33	30	32	30
484	58	69	51	62	49
664	87	111	75	112	81
1004	115	156	108	167	125
1231	120	172	115	179	142
1372	142	203	139	209	174
1582	145	203	140	214	177

3 Model

3.1 Description

Pinheiro and Bates (2000, pp. 356, 360) used the following mixed-effects logistic model to analyze the orange tree growth data,

$$y_{ij} = \frac{\phi_1 + b_i}{1 + \exp[-(x_{ij} - \phi_2)/\phi_3]} + \epsilon_{ij}, \qquad \epsilon_{ij} \sim N(0, \sigma^2)$$

where y_{ij} is the circumference of tree i at time j, x is the age in days, ϕ_1 is the asymptote (maximum circumference), ϕ_2 is the inflection point (age when trees have reached half maximum circumference), and ϕ_3 is a scale parameter (days it takes to grow from 50% to 73% of maximum circumference[1]).

The asymptote varies between trees as a random effect:

$$b_i \sim N(0, \sigma_b^2)$$

The traces in Figure 1 can be used to get sensible starting values for the three parameters and set the initial asymptote $\phi_1 = 200$, the inflection point $\phi_2 = 800$, and the scale parameter $\phi_3 = 400$.

Symmetric confidence limits around the ϕ regression coefficients are constructed by multiplying the estimated standard error with the normal quantile z:

$$\mathrm{CI}_{\hat{\phi}_i} = \hat{\phi}_i \pm z_{\alpha/2}\, \widehat{\mathrm{SE}}_{\hat{\phi}_i}$$

Asymmetric confidence limits around the σ and σ_b standard deviations are based on the standard error of the log-transformed parameters:

$$\mathrm{CI}_{\hat{\sigma}} = \exp\Big(\log \hat{\sigma} \pm z_{\alpha/2}\, \widehat{\mathrm{SE}}_{\log \hat{\sigma}}\Big)$$

3.2 R

Implementing the model in R is easy after loading the `nlme` package:

```r
library("nlme")
fm <- nlme(circumference~phi1/(1+exp(-(age-phi2)/phi3)),
           fixed=phi1+phi2+phi3~1, random=phi1~1|Tree,
           data=Orange, start=c(phi1=200,phi2=800,phi3=400))
```

There is also a "self-starting" `SSlogis()` function in R, specifically for fitting logistic models, but the above is a basic general approach for any nonlinear mixed-effects model. (Using the `SSlogis()` function speeds up the fit by about 15%, because in addition to providing initial conditions `SSlogis()` also returns an analytically computed gradient of the sum-of-squares function.)

[1]The 73% comes from $1/[1+\exp(-1)]$.

3.3 ADMB

In ADMB, the data and model are in two different text files, and the initial parameter values are in a third text file.

The data are in a file called `ora.dat`,

```
# Number of trees (M)
5

# Number of ages (n)
7

# Age (x, in days)
118 484 664 1004 1231 1372 1582

# Circumference (y, in mm)
30 58  87 115 120 142 145
33 69 111 156 172 203 203
30 51  75 108 115 139 140
32 62 112 167 179 209 214
30 49  81 125 142 174 177
```

the model code is in a file called `ora.tpl`,

```
DATA_SECTION
  init_int M
  init_int n
  init_vector x(1,n)
  init_matrix y(1,M,1,n)

PARAMETER_SECTION
  init_number phi1
  init_number phi2
  init_number phi3
  init_number logSigma
  init_number logSigmaB(3)         // estimated in phase 3
  random_effects_vector b(1,M,2)   // estimated in phase 2
  sdreport_number sigma
  sdreport_number sigmaB
  matrix yfit(1,M,1,n)
  number RSS
  objective_function_value f

PROCEDURE_SECTION
  sigma = exp(logSigma);
  sigmaB = exp(logSigmaB);
  for (int i=1; i<=M; i++)
  {
    for (int j=1; j<=n; j++)
    {
      yfit(i,j) = (phi1 + b(i)) / (1.0 + exp(-(x(j)-phi2)/phi3));
    }
  }
  RSS = sum(square(y-yfit));
  f = 0.5*M*n*log(2.0*M_PI) + M*n*logSigma + RSS/(2.0*square(sigma));
  f += 0.5*M*log(2.0*M_PI) + M*logSigmaB + sum(square(b))/(2.0*square(sigmaB));

GLOBALS_SECTION
```

```
   #define REPORT(object) report << "# " #object "\n" << object << endl;

REPORT_SECTION
   REPORT(yfit);
```

and the initial parameter values are in a file called `ora.pin`:

```
# phi1
200

# phi2
800

# phi3
400

# logSigma
1

# logSigmaB
0

# b (1,M)
0 0 0 0 0
```

This particular ADMB implementation is simple, not taking advantage of efficiency improvements such as separable functions and estimating unscaled random effects (Skaug and Fournier, 2006; Fournier et al., 2012). For more advanced usage, see the `wildflowers`, `owl`, and `theta` examples at `https://groups.nceas.ucsb.edu/non-linear-modeling/projects`.

The model is compiled with the shell command

```
admb -r ora
```

and then run:

```
ora
```

It can also be compiled and run from within R using the `R2admb` package:

```
library("R2admb")
setup_admb()  ## establish path etc. for ADMB
```

```
compile_admb("ora",re=TRUE)
run_admb("ora")
```

3.4 BUGS

The BUGS model is also fairly simple in this case.

```
model {
  for (i in 1:K) {
    for (j in 1:n) {
      Y[j, i] ~ dnorm(eta[j, i], tauC)
      eta[j, i] <- phi1[i] / (1 + exp(-(x[j]-phi2)/phi3))
    }
    ## random effect of i^th tree
    phi1[i] ~ dnorm(mu1, tau1)
  }
  ## priors
  tauC ~ dgamma(1.0E-3, 1.0E-3)
  logSigma <- -0.5*log(tauC)
  phi2 ~ dnorm(0, 1.0E-4)
  phi3 ~ dnorm(0, 1.0E-4)
  mu1~ dnorm(0, 1.0E-4)
  tau1~ dgamma(1.0E-3, 1.0E-3)
}
```

It loops over trees `i` and observations `j`, computing the expected size (`eta[j,i]`) incorporating the random effect of the `i`th tree (`phi[i]`), and specifying that `phi1[i]` is normally distributed. The rest of the code specifies the priors.

The model can be run from within R using the `R2jags` package.

As written the BUGS code requires the data to be arranged in *wide* format as in Table 1 (and as the ADMB input is structured), rather than in *long* format of the `Orange` variable (preferred by `nlme`: we use the `reshape` function to get a table whose first column is the age, with the other columns corresponding to measurements from specific trees.

```
OrangeTab <- reshape(Orange,timevar="Tree",direction="wide",idvar="age")
```

```
require("R2jags")
BUGSData<-list(n = 7, K = 5, x = OrangeTab[,1], Y = OrangeTab[,-1])
set.seed(1001) ## set RNG seed for R
inits<-list(phi1=200, phi2=800, phi3=400,tauC = 0.1, tau1=1,
            .RNG.name="base::Wichmann-Hill",  ## set RNG seed/type for JAGS
            .RNG.seed=round(runif(1)*1000000))
```

```
tfit_jags <- jags(model="../BUGS/OrangeTreeRE2_bugs.txt",
                  data=BUGSData,
                  parameters.to.save=c("phi1","phi2","phi3",
                                       "tau1","logSigmaB","logSigma","b1"),
                  n.chains=1,
                  inits=list(inits),
                  progress.bar="none")
```

3.5 Simulations

10 000 datasets are generated (Table 2) and the R and ADMB model implementations are evaluated in terms of computational speed, convergence, bias, and coverage probability.

Table 2. Parameter values used to simulate datasets for the second part of this study.

Parameter	Value
ϕ_1	191.05
ϕ_2	722.54
ϕ_3	344.15
σ	7.85
σ_b	31.48

4 Results

4.1 Model fit to original data

4.1.1 R

The R command `summary(fm)` summarizes the model fit:

```
## Nonlinear mixed-effects model fit by maximum likelihood
##   Model: circumference ~ phi1/(1 + exp(-(age - phi2)/phi3))
##  Data: Orange
##     AIC   BIC logLik
##   273.2 280.9 -131.6
##
## Random effects:
##  Formula: phi1 ~ 1 | Tree
##          phi1 Residual
## StdDev: 31.48    7.846
##
## Fixed effects: phi1 + phi2 + phi3 ~ 1
##      Value Std.Error DF t-value p-value
## phi1 191.0     16.15 28   11.83       0
## phi2 722.5     35.15 28   20.56       0
## phi3 344.2     27.15 28   12.68       0
##  Correlation:
##      phi1  phi2
## phi2 0.375
## phi3 0.354 0.755
```

```
##
## Standardized Within-Group Residuals:
##     Min      Q1     Med      Q3     Max
## -1.9148 -0.5351  0.1436  0.7310  1.6615
##
## Number of Observations: 35
## Number of Groups: 5
```

and `ranef(fm)` shows the random effects:

```
##      phi1
## 3 -37.001
## 1 -29.405
## 5  -5.178
## 2  31.565
## 4  40.020
```

4.1.2 ADMB

The ADMB executable produces several output files.

The negative log-likelihood and parameter estimates are in a file called `ora.par`,

```
# Number of parameters = 5  Objective function value = 131.572  Maximum gradient component = 1.39935e-05
# phi1:
192.053262305
# phi2:
727.906256776
# phi3:
348.073016244
# logSigma:
2.05962317341
# logSigmaB:
3.45462142782
# b:
 -29.5622333866 31.7278487931 -37.1935581226 40.2245464842 -5.19720353321
```

and standard errors and correlations are in a file called `ora.cor`:

```
 The logarithm of the determinant of the hessian = -11.7684
 index   name        value      std.dev       1       2       3       4       5       6
7
     1   phi1       1.9205e+02 1.5658e+01   1.0000
     2   phi2       7.2791e+02 3.5249e+01   0.3937  1.0000
     3   phi3       3.4807e+02 2.7080e+01   0.3732  0.7747  1.0000
     4   logSigma   2.0596e+00 1.2910e-01   0.0002  0.0001  0.0010  1.0000
     5   logSigmaB  3.4548e+00 3.2425e-01   0.0414  0.0987  0.0913 -0.0079  1.0000
     6   b         -2.9562e+01 1.4739e+01  -0.8957 -0.0671 -0.0601  0.0104 -0.0323  1.0000
     7   b          3.1728e+01 1.4742e+01  -0.8391  0.0677  0.0642 -0.0111  0.0344  0.8982
1.0000
```

```
     8   b         -3.7194e+01 1.4759e+01  -0.9007 -0.0808 -0.0749  0.0130 -0.0403  0.9093
0.8957  1.0000
     9   b          4.0225e+01 1.4767e+01  -0.8301  0.0864  0.0796 -0.0141  0.0435  0.8951
0.9093  0.8923  1.0000
    10   b         -5.1972e+00 1.4700e+01  -0.8734 -0.0064 -0.0089  0.0018 -0.0053  0.9070
0.9055  0.9059  0.9037  1.0000
    11   sigma      7.8430e+00 1.0125e+00   0.0002  0.0001  0.0010  1.0000 -0.0079  0.0104 -0.0111
0.0130 -0.0141  0.0018  1.0000
    12   sigmaB     3.1653e+01 1.0264e+01   0.0414  0.0987  0.0913 -0.0079  1.0000 -0.0323
0.0344 -0.0403  0.0435 -0.0053 -0.0079  1.0000
```

The predicted values, which we reported in the `REPORT_SECTION` of the TPL file, appear in a file called `ora.rep`:

```
# yfit
 24.0105 53.8901 73.8081 111.878 131.501 140.422 149.628
 33.0671 74.2169 101.648 154.078 181.103 193.387 206.066
 22.8829 51.3592 70.3417 106.624 125.326 133.827 142.601
 34.3226 77.0349 105.507 159.928 187.979 200.73 213.89
 27.6108 61.9708 84.8753 128.654 151.22 161.477 172.064
```

Alternatively, the point estimates and standard errors (without correlations) can be found in a file called `ora.std`.

The `R2admb` package can be useful for reading ADMB output back into R for plotting and analysis.

```
admb_res <- read_admb("ora")
```

creates an R object that contains all of the information from the ADMB fit.

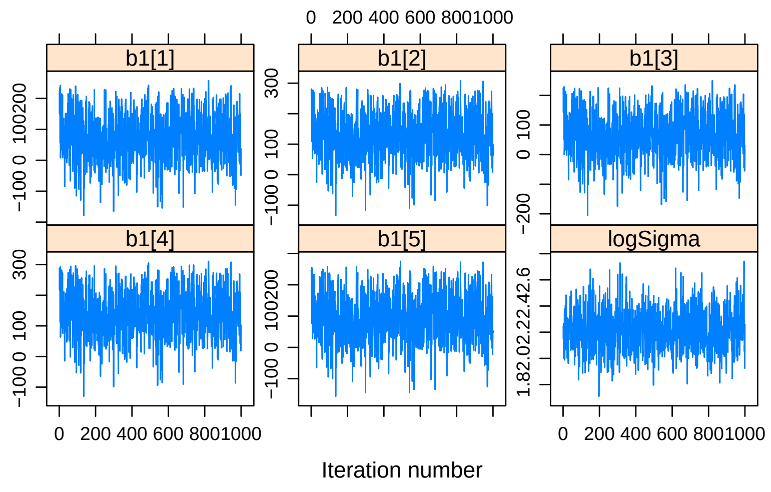
4.1.3 BUGS

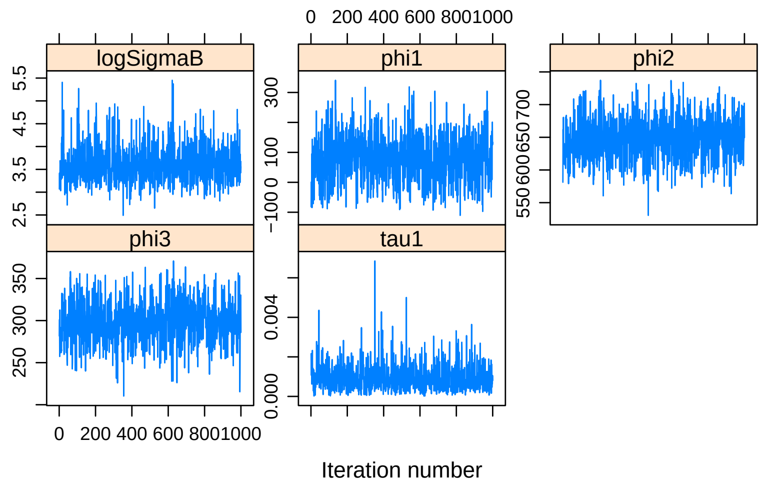
The default `print` method for `rjags` objects gives basic information:

```
## Inference for Bugs model at "../BUGS/OrangeTreeRE2_bugs.txt", fit using jags,
##  1 chains, each with 2000 iterations (first 1000 discarded)
##  n.sims = 1000 iterations saved
##           mu.vect sd.vect    2.5%     25%     50%     75%   97.5%
## b1[1]      73.503  76.997 -80.704  18.105  78.218 124.886 215.508
## b1[2]     130.267  77.207 -26.092  75.413 133.063 181.531 274.555
## b1[3]      66.238  76.948 -88.827  11.210  68.783 117.649 209.075
## b1[4]     138.013  77.203 -15.468  84.113 141.277 190.230 280.973
## b1[5]      95.737  77.221 -61.334  41.405  99.219 146.827 239.180
## logSigma    2.211   0.159   1.912   2.100   2.201   2.312   2.546
## logSigmaB   3.607   0.402   2.977   3.329   3.563   3.818   4.584
## phi1       77.837  77.182 -66.585  26.718  73.422 131.980 237.564
## phi2      648.130  30.958 589.151 628.155 648.011 668.089 711.639
## phi3       297.474   24.834 249.736 282.411 296.538 312.911 348.204
## tau1         0.001    0.001   0.000   0.000   0.001   0.001   0.003
## deviance   254.446    7.309 242.844 248.946 253.469 258.521 271.596
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 26.7 and DIC = 281.2
## DIC is an estimate of expected predictive error (lower deviance is better).
```

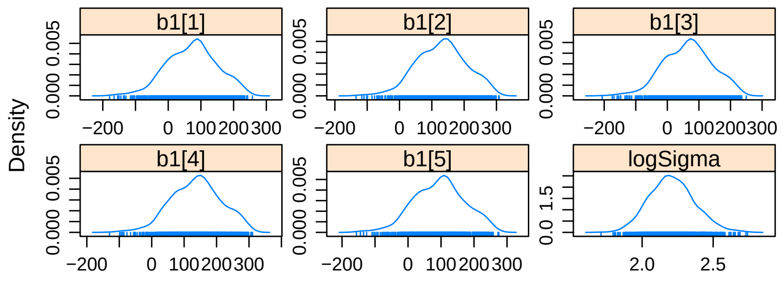
We can use the emdbook and coda packages to transform and plot the results (trace and density plots):

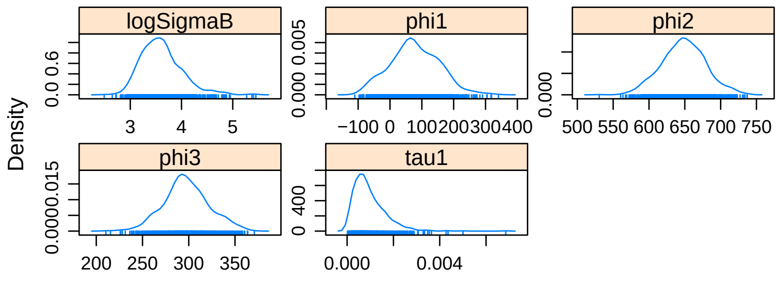
```
library(emdbook)
m <- as.mcmc.bugs(tfit_jags$BUGSoutput)
library(coda)
xyplot(m[,colnames(m)!="deviance"],layout=c(3,2),as.table=TRUE)
```





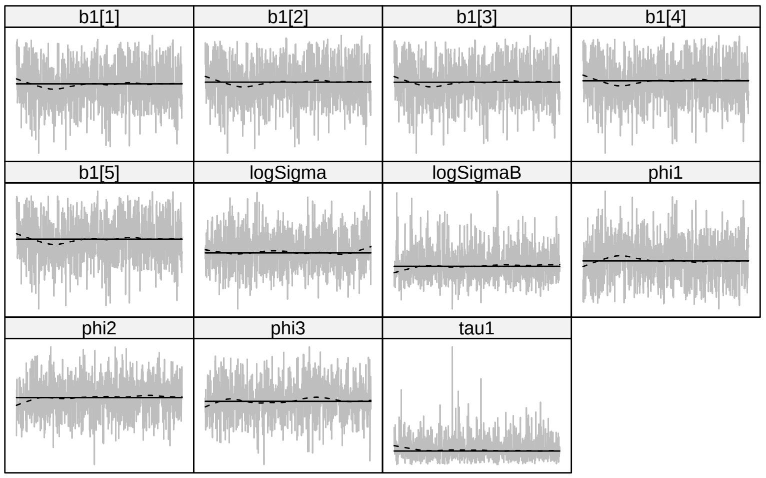
```
densityplot(m[,colnames(m)!="deviance"],layout=c(3,2),as.table=TRUE)
```





The `scapeMCMC` package gives alternative (prettier) plots.

```r
library(scapeMCMC)
plotTrace(m[,colnames(m)!="deviance"])
```



```r
plotDens(m[,colnames(m)!="deviance"])
```

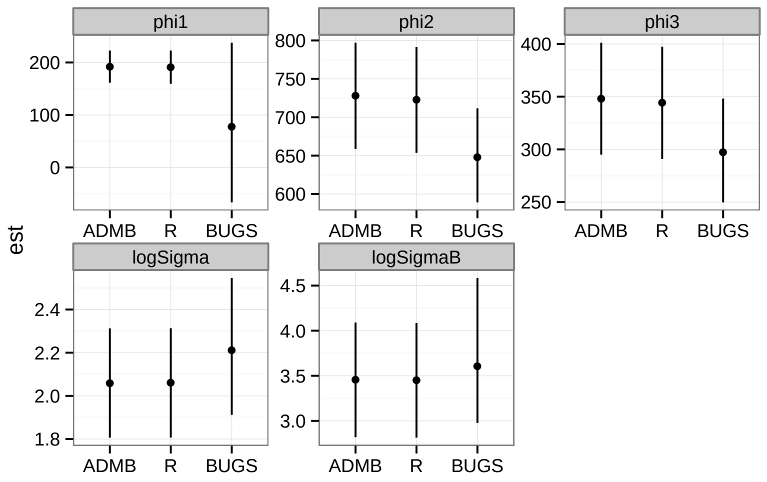
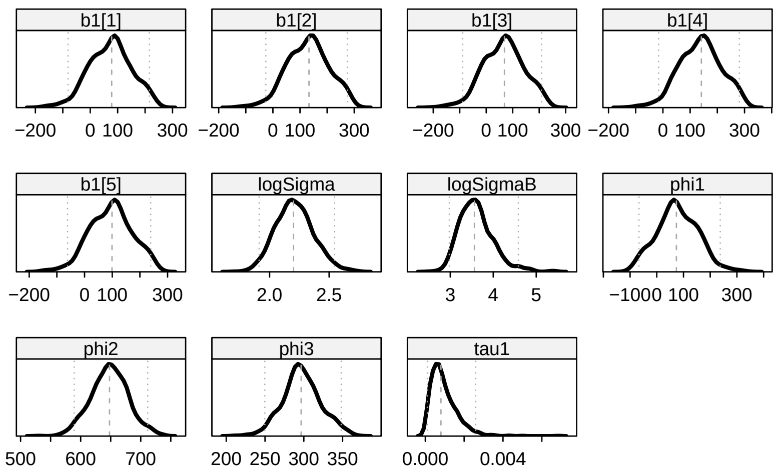


Figure 2. Comparison of parameter estimates from different tools. For ADMB and R, points show maximum likelihood estimates and error bars show 95% confidence intervals; for BUGS, points show posterior means and error bars show 95% Bayesian credible intervals.



4.1.4 Comparison

The ADMB and R estimates are extremely similar: the BUGS fits are slightly different, in part because BUGS is using the posterior mean rather than the posterior mode as a point estimate.

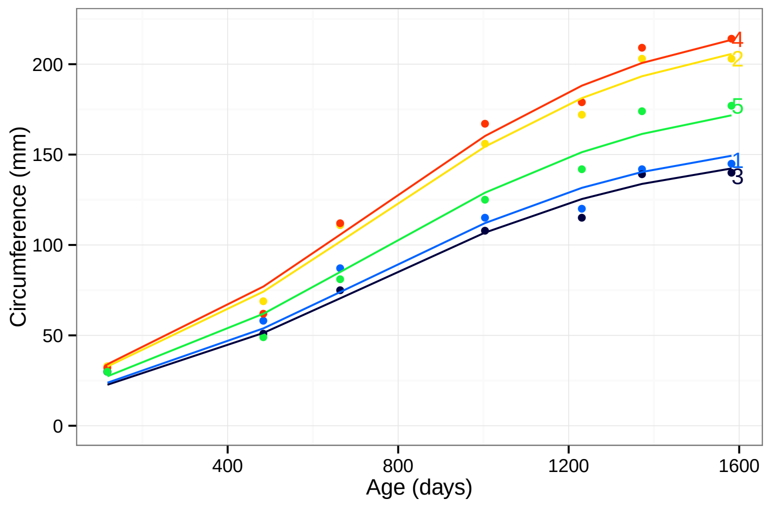


Figure 3. Model fit to the data (the fitted values from R and ADMB are indistinguishable).

4.2 Performance with simulated data

Speed

Fitting 10 000 models took 7:37 minutes in R (0.05 sec/run) and 56:37 minutes in ADMB (0.34 sec/run) on an old laptop computer, so the model runs seven times faster in R than in ADMB. If a similar model was to be used in a computationally intensive simulation study, it would be worthwhile to take advantage of efficiency improvements such as separable functions and estimating unscaled random effects in ADMB (Skaug and Fournier, 2006; Fournier et al., 2012).

Convergence

Out of 10 000 simulated datasets, 13 model runs did not converge, both in R and ADMB. In R, non-convergence was identified using the `intervals` function, where 13 models returned either an error or an upper σ_b confidence limit greater than 10^{10}. Likewise, non-convergence in ADMB was identified from the standard error of $\log \hat{\sigma}_b$ being either `NA` or greater than 10^{10}. This occurred in the same set of 13 simulated datasets.

To circumvent the problem of non-convergence, 13 additional simulated datasets were generated. The subsequent analysis of bias and coverage probability is therefore based on 10 000 converged simulations.

Bias

In both R and ADMB, the ϕ parameter estimates are unbiased, but σ and σ_b are underestimated with a relative bias of around -0.04 and -0.19 (Table 3, Figure 4). In terms of bias, the difference between R and ADMB is negligible.

Table 3. Median of 10 000 parameter estimates compared to the true parameter values used from the operating model. The relative bias is calculated as median$((\hat{\theta}-\theta)/\theta)$.

		R		ADMB	
	True	Median	Bias	Median	Bias
ϕ_1	191.05	190.78	0.00	191.59	0.00
ϕ_2	722.54	718.62	-0.01	723.31	0.00
ϕ_3	344.15	340.72	-0.01	344.04	0.00
σ	7.85	7.51	-0.04	7.51	-0.04
σ_b	31.48	25.52	-0.19	25.66	-0.18

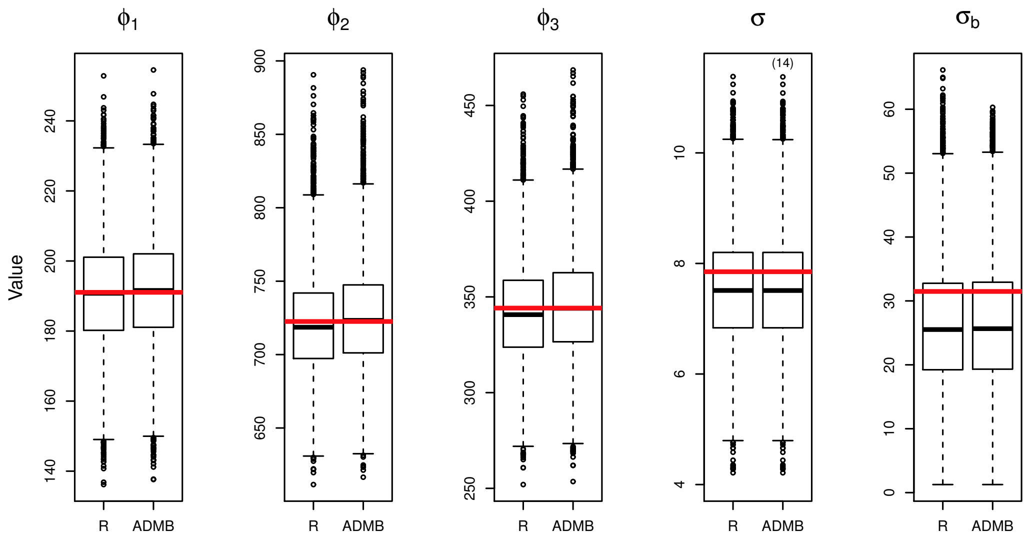


Figure 4. Distribution of point estimates (Tukey boxplots) compared to the true parameter value (red line) of each parameter. Fourteen σ estimates are outside the y-axis limits in the case of ADMB.

Coverage probability

Analysis of 10 000 confidence intervals the 90% confidence level shows that both R and ADMB generate confidence intervals that cover the true parameter less than 90% of the time (Table 4). The performance is poor for σ_b and ϕ_1, with coverage probability around 78% and 82%, but considerably better for the other parameters.

Table 4. Coverage probability of 90% confidence intervals generated using R and ADMB. Ideally, the coverage probability at this confidence level should be 0.900 for every parameter.

	R	ADMB
ϕ_1	0.824	0.813
ϕ_2	0.884	0.879
ϕ_3	0.886	0.877
σ	0.861	0.860
σ_b	0.783	0.787

Analysis of confidence levels ranging from 0 to 99% shows the same trends (Figure 5). Both R and ADMB generate confidence intervals that are too narrow, especially for σ_b and ϕ_1. Overall, R and ADMB show similar performance in terms of coverage probability: R performs slightly better for the ϕ parameters, but ADMB performs slightly better for the problematic σ_b parameter.

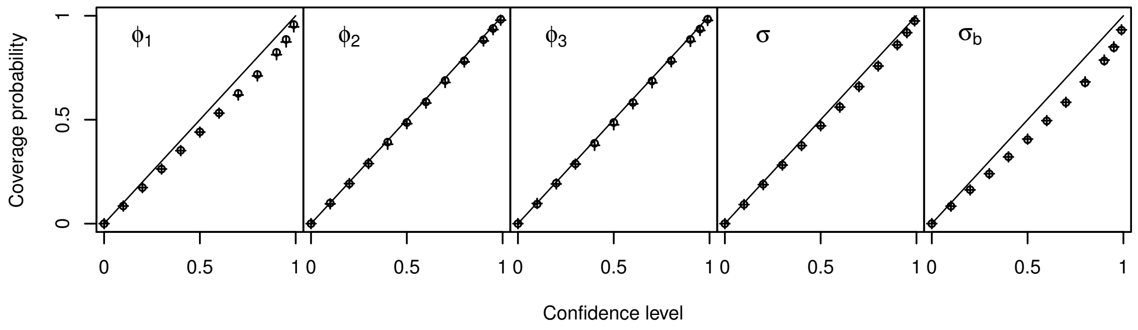


Figure 5. Coverage probability for confidence intervals evaluated at several confidence levels (0, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 99%). Each panel shows the performance of R (circle) and ADMB (cross) for a given parameter.

5 Discussion

We have demonstrated how simple nonlinear mixed-effects models can be fitted in R, AD Model Builder, and BUGS.

For this basic example, R and ADMB show similar estimation performance on the whole. To fit a mixed-effects logistic growth model to the orange tree data, it is easier and faster to use the `nlme` package in R, yielding similar results as ADMB. One possible reason to use ADMB for analyzing this dataset might be to explore other modelling options (statistical assumptions and methods) that are not provided by the `nlme` function in R.

References

Fournier, D. A., H. J. Skaug, J. Ancheta, J. Ianelli, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert (2012). AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods & Software 27*(2), 233–249.

Madsen, H. and P. Thyregod (2010). *Introduction to General and Generalized Linear Models.* Boca Raton, FL: CRC Press.

Pinheiro, J. C. and D. M. Bates (2000). *Mixed-effects models in S and S-PLUS.* New York: Springer.

Skaug, H. and D. Fournier (2006). Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models. *Computational Statistics & Data Analysis 51*(2), 699–709.