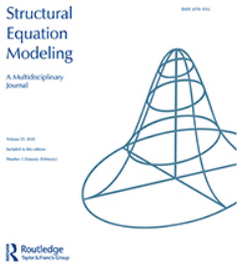
Routledge
Taylor & Francis Group



Structural Equation Modeling: A Multidisciplinary Journal

ISSN: 1070-5511 (Print) 1532-8007 (Online) Journal homepage: http://www.tandfonline.com/loi/hsem20

Nonlinear Mixed-Effects Modeling Programs in R

Gabriela Stegmann, Ross Jacobucci, Jeffrey R. Harring & Kevin J. Grimm

To cite this article: Gabriela Stegmann, Ross Jacobucci, Jeffrey R. Harring & Kevin J. Grimm (2018) Nonlinear Mixed-Effects Modeling Programs in R, Structural Equation Modeling: A Multidisciplinary Journal, 25:1, 160-165, DOI: 10.1080/10705511.2017.1396187

To link to this article: https://doi.org/10.1080/10705511.2017.1396187

Published online: 08 Dec 2017.

Structural Equation Modeling: A Multidisciplinary Journal, 25: 160–165, 2018

ISSN: 1070-5511 print / 1532-8007 online
DOI: https://doi.org/10.1080/10705511.2017.1396187

Routledge
Taylor & Francis Group

Check for updates

SOFTWARE REVIEW

Nonlinear Mixed-Effects Modeling Programs in R

Gabriela Stegmann,[1] Ross Jacobucci,[2] Jeffrey R. Harring,[3] and Kevin J. Grimm[1]

[1]*Arizona State University*
[2]*University of Notre Dame*
[3]*University of Maryland*

In this software review, we provide a brief overview of four R functions to estimate nonlinear mixed-effects programs: nlme (linear and nonlinear mixed-effects model), nlmer (from the lme4 package, linear mixed-effects models using Eigen and S4), saemix (stochastic approximation expectation maximization), and brms (Bayesian regression models using Stan). We briefly describe the approaches used, provide a sample code, and highlight strengths and weaknesses of each.

Keywords: nonlinear mixed-effects models, R software, mixed-effects model functions in R, mixed-effects modeling programs in R

In R (R Core Team, 2016), there exist a multitude of packages to perform each statistical analysis. In this article, we review four existing functions and packages in R for fitting nonlinear mixed-effects models, which are commonly used to model individual change patterns that follow inherently nonlinear functions (Grimm, Ram, & Hamagami, 2011). There are different approaches to estimating nonlinear mixed-effects models and the available packages in R cover a range of these options. The four packages described herein include `nlme` (linear and nonlinear mixed-effects model; Pinheiro, Bates, DebRoy, & Sarkar, 2017), `nlmer` (from the `lme4` package, linear mixed-effects models using Eigen and S4; Bates, Maechler, Bolker, & Walker, 2015), `saemix` (stochastic approximation expectation maximization; Comets, Lavenu, & Lavielle, 2017), and `brms` (Bayesian regression models using Stan; Buerkner, 2016). We first provide a brief overview of nonlinear mixed-effects models, followed by a description of the illustrative data used throughout the article, a description of the nonlinear mixed-effects modeling packages, and a brief comparison of the four packages.

Correspondence should be addressed to Gabriela Stegmann, Arizona State University, Department of Psychology, PO Box 871104, Tempe, AZ 85287. E-mail: gabriela.stegmann@asu.edu

NONLINEAR MIXED-EFFECTS MODELS

Mixed-effects models are commonly used when data consist of clusters of observations. The fixed-effects parameters describe the general patterns of the data and random-effects parameters describe specific clusters. If the model is nonlinear in the parameters, it is called a nonlinear mixed-effects model (Davidian & Giltinan, 2003). Growth data where the change trajectory is nonlinear is a common type of data modeled with nonlinear mixed-effect models, with the subject treated as a cluster and the repeated measurements as the individual observations nested within the cluster. In such a situation, the fixed effects describe the change trajectory of the population, whereas the random effects reflect variability between individuals (Bryk & Raudenbush, 1992; Laird & Ware, 1982). The general model for a mixed-effects model is

$$y_{it} = f(z_{it}, b_{ki}) + e_{it}$$

with

$$b_{ki} = \beta_k + r_{ki}, \quad (1)$$

where y_{it} is the outcome of interest for individual $i = 1, \ldots n$ measured repeatedly at times $t = 1, \ldots, T_i$, following the functional form f, which depends on the parameters

b_{ki} for $k = 1, \ldots, k$ and z_{it} is the value of the timing metric. The random effects, r_{ki} and e_{it}, have means of 0, and are typically assumed to follow a normal distribution with their respective covariance matrices (Pinheiro & Bates, 2000).

Illustrative Data

Throughout the article, as we describe each of the R packages, we use empirical data from the Berkeley Growth Study (BGS; $N = 83$) to illustrate their use. This BGS was initiated by Bayley in 1928 to track individual changes in cognitive, motor, and physical abilities during the first year of life (Jones, Bayley, Macfarlane, & Hoznik, 1971). Here, we make use of longitudinal data of the children's height obtained from their first 3 years of life. Figure 1 contains a longitudinal plot of a random sample of $n = 40$ subjects, where the nonlinearity in the individual trajectories is apparent. For our illustration, we model the children's height as a function of age using a form of the Jenss–Bayley growth model (Grimm, Ram, & Estabrook, 2017; Jenss & Bayley, 1937), which can be written as

$$y_{it} = b_{1i} + b_{2i} * \left(\frac{age_{it}}{12}\right) + b_{3i}\left(exp\left(\gamma * \left(\frac{age_{it}}{12}\right)\right) - 1\right) + e_{it}$$

where

$$b_{ki} = \beta_k + r_{ki} \quad (2)$$

for parameter with $k = 1$, 2, and 3, such that y_{it} is the height of the child i at time t, b_{1i} is the child's intercept, which is the height at birth, b_{2i} is the child's slope of the linear asymptote, b_{3i} is the vertical distance between the actual intercept and the intercept of the linear asymptote for the child i, and $\exp(\gamma)$ is the ratio of acceleration of growth from one year to the next, and age_{it} is the child's age in months.

R PACKAGES FOR NONLINEAR MIXED-EFFECTS MODELS

In this section, overviews of `nlme()`, `nlmer()`, `saemix()`, and `brms()` are given. We discuss the package, demonstrate how the relevant function is used in R, and show results obtained when applying the Jenss–Bayley model to the BGS height data.

`nlme`

The `nlme` function contained in the `nlme` package of R (Pinheiro et al., 2017) fits user-specified nonlinear mixed-effects models using the Lindstrom and Bates (1990) method. This algorithm alternates between two steps: a penalized nonlinear least squares (PNLS) step and a linear mixed effects (LME) step, both of which are iterative algorithms. In the PNLS step, the current estimate of the covariance matrix of the random effects is fixed, and the estimates of β_k and r_{ki} are obtained by minimizing the PNLS function. The LME step uses the current estimates of β_k and r_{ki} to update the estimate of the covariance matrix of the random effects based on a first-order Taylor series expansion of r_{ki} around the current estimates of b_{ki}. During the initial stage of the LME step, the parameter estimates are updated using the expectation-maximization (EM) algorithm, and then after obtaining a better estimate of the parameters, the optimization switches to the Newton–Raphson algorithm. Although the Newton–Raphson algorithm is a quadratic hill climbing algorithm and therefore converges more quickly than the EM algorithm (which converges in a linear fashion), the EM

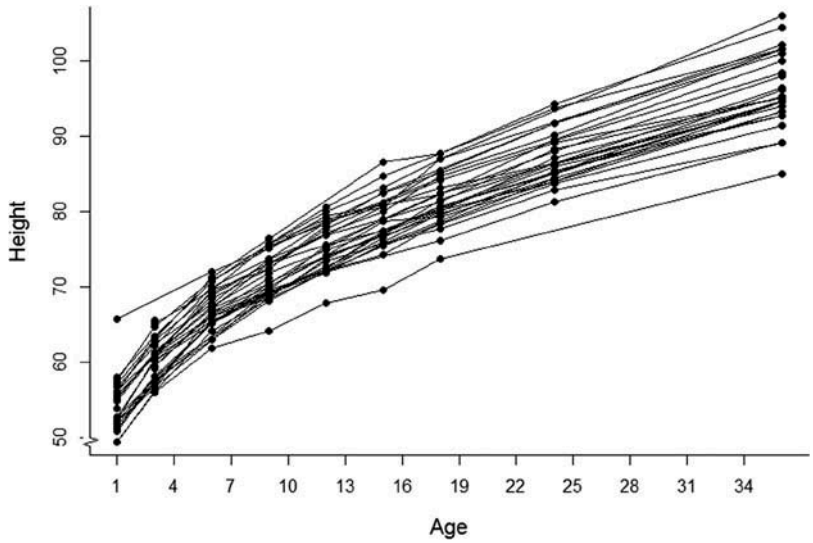


FIGURE 1 Spaghetti plot for a random sample for height as a function of age of 40 children.

algorithm does not suffer from poor starting values as the Newton–Raphson algorithm does, which makes this hybrid algorithm state of the art for its speed and sensitivity (see Lindstrom & Bates, 1988, 1990). The Lindstrom and Bates algorithm maximizes the log-likelihood by default, but the researcher can specify to maximize the restricted log-likelihood instead.

Code to fit the Jenss–Bayley mixed-effects model to the empirical data is shown here:

```
hght.jb.nlme <- nlme(hght~b_1i+b_2i*(age/12)+b_3i*
  (exp(gamma*(age/12))-1),
data = hght_long,
fixed = b_1i+b_2i+b_3i+gamma~1,
random = b_1i+b_2i+b_3i~1,
groups = ~ id,
start = c(50, 10, -18, -2),
na.action = na.omit)
summary(hght.jb.nlme)
```

We first specified the model, which contains the outcome of interest (`hght`) on the left side of the equation and the Jenss–Bayley function on the right side of the equation, separated by the tilde, following the common R regression notation. Under `fixed`, we specified the fixed effects of the equation; and under `random`, we specified the random effects of the equation. The cluster variable is then listed using the `group` statement and the name of the data set is entered under `data`. The researcher needs to specify the starting values for each fixed parameter under `start`, and the values we chose were based on the results of a previously fitted nonlinear regression model estimated using `nls()`. Starting values for the variance–covariance parameters can also be specified, although this was not done here. In `nlme`, the user can specify a heteroscedastic and correlated residual covariance structure; however, we used the default control conditions (see Pinheiro et al., 2017, for additional information), where a single Level-1 residual variance is specified.

nlmer

The function `nlmer` is contained within the `lme4` R package (Bates et al., 2015). The `lme4` package provides functions for fitting linear, generalized linear, and nonlinear mixed-effects models. The `nlmer` function fits nonlinear mixed-effects models using one of two approaches. The first approach uses adaptive Gaussian quadrature approximation, where the number of quadrature points can be greater than one when there is a single random effect, but for nonlinear mixed models (which tend to have more than one random effect), the number of quadrature points is one, which is equivalent to Laplace approximation (see Wolfinger, 1993, for details). This approach is the default for nonlinear mixed-effects models (`nAGQ = 1`). The second approach does not integrate out the random effects, such that these only influence the estimates of the fixed effects through their estimated conditional modes. This approach is requested by specifying `nAGQ = 0`, and it does not completely account for the randomness of the random effects. There are three steps to this optimization: (a) penalized iteratively reweighted least squares (PIRLS) to estimate the conditional modes of the random effects, (b) (approximately) integrate out the random effects about their conditional modes, and (c) nonlinear optimization of the objective function (i.e., the result of the integration). These steps are themselves iterated until convergence.

Code that we used to estimate the parameters of the Jenss–Bayley mixed-effects model is as follows:

```
jb_model <- function(age, b_1i, b_2i, b_3i, gamma)
b_1i+b_2i*(age/12)+b_3i*(exp(gamma*(age/12))-1)
jb_modelg <- deriv(body(jb_model),
namevec = c("b_1i", "b_2i", "b_3i", "gamma"),
function.arg = jb_model)
startsite <- c(b_1i = 50, b_2i = 10, b_3i = -18,
  gamma = -2)
jb.math.nlmer <- nlmer(hght ~ jb_modelg(age, b_1i,
  b_2i, b_3i, gamma) ~ (b_1i+b_2i+b_3i|id),
data = hght_long,
start = startsite)
summary(jb.math.nlmer)
```

When using the `nlmer` function, the formula is specified in three parts: the response on the left (`hght`), a nonlinear function in the middle, which also returns a gradient (`jb_modelg`), and the random effects with the grouping variable on the right (`b_1i+b_2i+b_3i|id`). Although `nlmer` has some built-in nonlinear functions that the researcher can choose from (as does `nlme`), the user can also specify this function using the `deriv()` function to estimate a user-defined nonlinear function. The researcher also needs to specify the data set under data and the starting values for the fixed effects under `start`. The default method of estimation is `nAGQ = 1`, which estimates the parameters using Laplace approximation.

saemix

The stochastic approximation expectation maximization (SAEM) algorithm estimates the parameters of nonlinear mixed-effects models by using a stochastic approximation of the likelihood using a modification of the EM algorithm (Dempster, Laird, & Rubin, 1977), as proposed by Kuhn and Lavielle (2005). The stochastic optimizer is used to overcome the difficulty of estimating parameters of nonlinear functions. As described by Comets et al. (2017), in the EM algorithm used to estimate the parameters of nonlinear mixed-effects models, in the E-step the conditional expectation of the log-likelihood given the current estimate of the parameters is computed, whereas in the M-step the parameter estimates that maximize the log-likelihood

function are found. In nonlinear models, however, the E-step does not have a closed solution and needs to be approximated. SAEM replaces the E-step with a stochastic procedure, such that individual parameter estimates $b_{ki}^{(m)}$ are simulated based on the conditional distribution given the current parameter estimates at iteration m using a Markov chain Monte Carlo. Then, using a stochastic approximation, the likelihood function is approximated. In the M-step, the parameter estimates that maximize the likelihood function at iteration m are obtained. The parameter λ_m must be between 0 and 1 and controls the convergence rate (the closer to 0, the faster the convergence).

During the initial m steps, λ_m is set at 1, such that the parameter space is explored. During the final iterations, λ_m moves closer to 0, which allows the estimator to converge. Because the log-likelihood is not directly estimated during the parameter estimation process, it can be estimated through linearization (similar to Lindstrom & Bates, 1990), importance sampling, and Gaussian quadrature (which `saemix()` uses a default of 12 nodes) after the parameters have been estimated (Comets et al., 2017). For more details on this procedure, the reader is referred to Comets et al. (2017).

Code to estimate the Jenss–Bayley mixed-effects model using the `saemix()` function is shown here:

```
jb <-saemixData(name.data = hgt,
name.group = "id",
name.predictors = "age",
name.response = "hgt",
units = list(x = "age",y = "height"))
jb.model <-function(psi, id, xidep){
age <- xidep[,1]
b1 <- psi[id,1]
  b2 <- psi[id,2]
b3 <- psi[id,3]
gam <- psi[id,4]
resp <- b1+b2*(age/12)+b3*(exp(gam*(age/12))-1)
return(resp)
}
saemix.model<-saemixModel(model = jb.model,
psi0 = matrix(c(50,9,-17,-2), ncol = 4, byrow = TRUE,
dimnames = list(NULL, c("b1","b2","b3","gam"))),
transform.par = c(0,0,0,0),
fixed.estim = c(1,1,1,1),
covariance.model = matrix(c(1,1,1,0,
1,1,1,0,
1,1,1,0,
0,0,0,0),
ncol = 4,byrow = TRUE))
opt <- list(seed = 94352514, save = FALSE, save.
  graphs = FALSE)
jb.fit <-saemix(saemix.model,jb,opt)
print(jb.fit)
```

The `saemix()` function requires two arguments: the structural model in `saemixModel()` and the data in `saemixData()`. In the `saemixData()` function, we specify the name of the data set, the cluster variable, the predictor, and the response. In the `units` argument within `saemixData` the user specifies the predictor x, the response y, and optional covariates. In the `saemixModel()` function, the two mandatory arguments are `model` (used to compute the structural model) and `psi0` (the initial estimates of the fixed parameters). Other arguments are optional. Within the `saemixModel` function, the argument `transform.par` specified a normal distribution of each parameter (which can also be specified to be log-normal, probit, or logit), `fixed.estim` specified that the parameters should be estimated (as opposed to fixed at their initial estimate), and the `covariance.model` argument specified a constant variance for `b1`, `b2`, `b3`, and 0 variance for gam (because it is a fixed parameter). Finally, in the control argument of `saemix()`, we specify a seed, and opted not to save the results or diagnostics in a file by setting `save = FALSE` and `save.graphs = FALSE`. For the other control conditions, we used the defaults (Comet et al., 2017; for more information, refer to the R package `saemix` user's guide).

brms

The `brms` package (Buerkner, 2016) interfaces with the probabilistic programming language `Stan` (Stan Development Team, 2017) for the implementation of multilevel models. As the specification of full Bayesian models can be daunting, particularly for beginners, the `brms` package greatly simplifies both the model syntax and accessing summaries of the results.

In contrast to other freely available Bayesian estimation software, such as `JAGS` (Plummer, 2003) and `BUGS` (Lunn, Spiegelhalter, & Thomas, 2009; Zhang, 2014), `Stan` implements both Hamiltonian Monte Carlo (Duane, Kennedy, Pendleton, & Roweth, 1987; Neal, 2011) and its extension, the No-U-Turn Sampler (Hoffman & Gelman, 2014), both of which converge much quicker in the presence of high-dimensional models. As a result, this form of estimation seems particularly advantageous for nonlinear mixed-effects models, specifically when the model includes multiple latent variables (Hoffman & Gelman, 2014).

Code to specify the Jenss–Bayley mixed-effects model using `brms` is as follows:

```
f1 <- hght ~ b1i+b2i*(age/12)+b3i*(exp(gamma*(age/
  12))-1)
prior_1 <- c(set_prior("normal(50, 5)",
  nlpar = "b1i"),
set_prior("normal(9.3, 2)", nlpar = "b2i"),
set_prior("normal(-17, 3)", nlpar = "b3i"),
set_prior("normal(-2, 2)", nlpar = "gamma"))
form = bf(f1,nl = TRUE)+list(gamma~1,
b1i~(1|2|id),b2i~(1|2|id),b3i~(1|2|id))
n1_b <- brm(form,data = hght_long,prior = prior_1)
summary(n1_b)
```

First, a function was created with the Jenss–Bayley model predicting `hght` as a function of `age`. Priors were then set for each of the fixed effects using the `set_prior` statements, where we specified a normal distribution for each of the

parameter estimates. `Form` contains the function, the list of fixed effects and random effects. Note that the syntax is very similar to that of `lme4`. The random effects are specified using `(1 | 2 | id)` for each parameter that includes a random effect for intercept and is correlated with other random effects, where `id` is the cluster variable. If the model is nonlinear (`nl = TRUE`), priors must be specified for all fixed effects. Finally, in the `brm` statement, the `form, data`, and `priors` are entered to estimate the model.

For these analyses, we used the default of 2,000 iterations across four chains and random starting values. The estimates from the nonlinear regression model were used as informative priors for the fixed effects parameters. This resulted in a model that converged using the potential scale reduction factor (all values < 1.01; Gelman & Rubin, 1992), with each parameter having a minimum effective sample size of 500. Note that without informative priors the model did not achieve convergence.

ESTIMATES AND COMPUTATIONAL TIME

The log-likelihoods and parameter estimates from the Jenss–Bayley model fit to the longitudinal height data from the BGS are contained in Table 1. Overall, estimates from each of the four packages were relatively consistent. For all methods, $\beta_1 \approx 51$, indicating that the expected height at age 0 was 51 inches, $\beta_3 \approx -18$ is the average annual slope of the linear asymptote, $\beta_2 \approx 9$ is the average vertical distance between the actual intercept and the intercept of the linear asymptote, and $\gamma \approx -2$ indicates that the ratio of acceleration of growth per year was approximately $\exp(-2)$. In terms of computational time for each of the four approaches, the fastest method was `nlme`, which ran for 1.186 sec before converging; followed by `nlmer`, which ran for 4.481 sec; `saemix`, which ran for 6.508 sec; and `brms`, which ran for 221.143 sec before converging to a solution.

TABLE 1
Parameter Estimates (and Standard Errors) Obtained From Each Program

Parameter	*nlme*	*nlmer*	*saemix*	*brms*
β_1	51.05 (.52)	51.09 (.34)	51.18 (.34)	51.10 (.36)
β_2	9.31 (.41)	9.27 (.17)	9.30 (.17)	9.28 (.18)
β_3	−17.82 (.88)	−17.88 (.46)	−17.76 (.46)	−17.86 (.48)
γ	−2.09 (.23)	−2.06 (.07)	−2.05 (.07)	−2.07 (.08)
σ_{r1}	2.72	2.72	2.71	2.78 (.26)
σ_{r2}	.86	.86	.87	.89 (.11)
σ_{r3}	3.06	3.07	3.06	3.13 (.36)
σ_e	.82	.82	.82	.83 (.03)
$cor(r_1, r_2)$	.26	.26	.25	.23 (.14)
$cor(r_1, r_3)$	.58	.58	.58	.55 (.10)
$cor(r_2, r_3)$	.25	.25	.26	.22 (.16)
Log-likelihood	−989.12	−989.1	−1002.62[a] −989.58[b]	−1031.59[c]

Note. [a]Linearized log-likelihood. [b]Importance sampling log-likelihood. [c]Bridge sampling estimate of the log-marginal-likelihood.

EVALUATION

The `nlme` function uses the simplest syntax and was the fastest to converge. Pinheiro and Bates (2000) offered great documentation on this function in their book in addition to the user's manual, which is particularly beneficial to researchers who are new to nonlinear mixed-effects models estimation. It can handle multiple levels of variability and fit correlated and heteroscedastic error models. However, the Lindstrom and Bates (1990) algorithm used in this function gives less accurate results than the Laplace approximation, used in `nlmer`. Pinheiro and Bates (1995), in their simulation study, recommended that the Lindstrom and Bates algorithm (1990) should be used to obtain starting values for the adaptive Gaussian quadrature method because the adaptive quadrature method relies too heavily on very good starting values—especially for the variance–covariance components. The `nlmer` function offers adaptive Gaussian quadrature with one quadrature point for nonlinear mixed-effects models (Laplace approximation). The researcher has to define a function to obtain the derivatives of the model, which, even though it is easy to do, can bring difficulties if the gradient of the model is undefined at some point. A drawback is that although there is great documentation on the `lmer` function (for linear models), there appears to be less available documentation on the use of the `nlmer` function.

Although `saemix` was slower to converge, simulation studies by Comets et al. (2017) have shown substantially higher convergence rates than `nlme` and `nlmer`, especially as the models become more complex and with sparse sampling. The `saemix` function is documented in Comets et al. (2017), which offers a detailed description of the method, the package, a worked example with syntax, and a simulation study comparing its performance to `nlme` and `nlmer`. The `brms` function offers a Bayesian framework for estimating nonlinear mixed-effects models in which the researcher can incorporate prior knowledge about the parameter estimates. It was the slowest model to converge because of the use of the Hamiltonian Monte Carlo sampler algorithm, but offers flexibility of distributions (e.g., categorical, zero-inflated) and model specifications (e.g., correlated error structures, user-specified covariance matrices) that can be fit. The documentation offered by Buerkner (2016) describes the algorithms used, example models that can be specified, a worked example with syntax, and a comparison to `lme4` and other Bayesian-based nonlinear mixed-effects modeling programs implemented inside and outside of R. We note that the `saemix` and `brms` packages are recent developments in R, and more evaluation is needed to determine how well these packages estimate nonlinear mixed-effects models. However,

given the algorithms implemented in these packages, we expect them to perform well with the potential to achieve higher rates of convergence for nonlinear mixed-effects models when compared to the nonlinear estimation routines available in `lmer` and `nlme`.

FUNDING

This work was supported by National Science Foundation Grant REAL-1252463 awarded to the University of Virginia, David Grissmer (Principal Investigator), and Christopher Hulleman (Co-Principal Investigator).

REFERENCES

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, *67*(1), 1–48. doi:10.18637/jss.v067.i01

Bryk, A., & Raudenbush, S. (1992). *Hierarchical linear models for social and behavioral research*. Newbury Park, CA: Sage.

Buerkner, P. C. (2016). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, *80*, 1–28.

Comets, E., Lavenu, A., & Lavielle, M. (2017). Parameter estimation in nonlinear mixed effect models using saemix, an R implementation of the SAEM algorithm. *Journal of Statistical Software*, *80*(3), 1–41. doi:10.18637/jss.v080.i03

Davidian, M., & Giltinan, D. M. (2003). Nonlinear models for repeated measurement data: An overview and update. *Journal of Agirculture, Biological, and Environmental Statistics*, *8*, 387–419. doi:10.1198/1085711032697

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*(1), 1–38.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, *195*, 216–222. doi:10.1016/0370-2693(87)91197-X

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science, 7*, 457–472. doi:10.1214/ss/1177011136

Grimm, K., Ram, N., & Estabrook, R. (2017). *Growth modeling: Structural equation and multilevel modeling approaches*. New York: Guilford Publications, Inc.

Grimm, K. J., Ram, N., & Hamagami, F. (2011). Nonlinear growth curves in developmental research. *Child Development*, *82*, 1357–1371. Retrieved from https://doi.org/10.1016/j.immuni.2010.12.017

Hoffman, M. D., & Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, *15*(1), 1593–1623.

Jenss, R. M., & Bayley, N. (1937). A mathematical method for studying the growth of a child. *Human Biology*, *9*, 556–563.

Jones, M. C., Bayley, N., Macfarlane, J. W., & Honzik, M. H. (Eds.). (1971). *The course of human development: Selected papers from the longitudinal studies, Institute of Human Development, the University of California, Berkeley*. Waltham, MA: Xerox College Publishing.

Kuhn, E., & Lavielle, M. (2005). Maximum likelihood estimation in nonlinear mixed effects models. *Computational Statistics and Data Analysis*, *49*, 1020–1038. doi:10.1016/j.csda.2004.07.002

Laird, N. M., & Ware, J. H. (1982). Random-effects models for longitudinal data. *Biometrics*, *38*, 963–974. doi:10.2307/2529876

Lindstrom, M. J., & Bates, D. M. (1988). Newton–Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association, 83*, 1014–1022. doi:10.1080/01621459.1988.10478693

Lindstrom, M. J., & Bates, D. M. (1990). Nonlinear mixed effects models for repeated measures data. *Biometrics*, *46*, 673–687. doi:10.2307/2532087

Lunn, D., Spegelhalter, D., & Thomas, A. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, *28*(25), 3049–3067.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, & X.-L. Meng (Eds.), *Handbook of Markov chain Monte Carlo* (Vol. 2). New York, NY: CRC.

Pinheiro, J., Bates, D., DebRoy, S., & Sarkar, D., & R Core Team (2017). *nlme: Linear and nonlinear mixed effects models*. Retrieved from https://CRAN.R-project.org/package=nlme

Pinheiro, J. C., & Bates, D. M. (1995). Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics*, *4*, 12–35.

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

Plummer, M. (2003). JAGS: *A program for analysis of Bayesian graphical models using Gibbs sampling*. In Hornik, K., Leisch, F., & Zeileis, A. (Eds.). Proceedings from the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), Vienna, Austria.

R Core Team. (2016). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Stan Development Team. (2017). Stan: A C++ library for probability and sampling (Version 2.16.0). Retrieved from http://mc-stan.org/.

Wolfinger, R. D. (1993). Laplace's approximation for nonlinear mixed models. *Biometrika*, *80*, 791–795. doi:10.1093/biomet/80.4.791

Zhang, Z. (2014). WebBUGS: Conducting Bayesian Analysis online. *Journal of Statistical Software*, *61*(7), 1–30.