Title

stata.com

menl — Nonlinear mixed-effects regression

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
References Also see

Description

`menl` fits nonlinear mixed-effects models in which some or all fixed and random effects enter nonlinearly. These models are also known as multilevel nonlinear models or hierarchical nonlinear models. The overall error distribution of the nonlinear mixed-effects model is assumed to be Gaussian. Different covariance structures are provided to model random effects and to model heteroskedasticity and correlations within lowest-level groups.

Quick start

Nonlinear mixed-effects regression of `y` on `x1` and `x2` with random intercepts `B0` by `id`

```
menl y = {a}*(1-exp(-({b0}+{b1}*x1+{b2}*x2+{B0[id]})))
```

As above, but using the more efficient specification of the linear combination

```
menl y = {a}*(1-exp(-{xb: x1 x2 B0[id]}))
```

As above, but using `define()` to specify the linear combination

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id])
```

As above, but perform restricted maximum-likelihood estimation instead of the default maximum-likelihood estimation

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id]) reml
```

Specify your own initial values for fixed effects, but use the default expectation-maximization (EM) method to obtain initial values for random-effects parameters

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id])             ///
    initial({a} 1 {xb:x1} 1 {xb:x2} 0.5 {xb:_cons} 2, fixed)
```

Include random intercepts `A0` by `id` to allow parameter `a` to vary between levels of `id` and specify the `xb` suboption to indicate that `a:` contains a linear combination rather than a scalar parameter

```
menl y = {a:}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id])            ///
    define(a: A0[id], xb)
```

Include a random slope on continuous variable `x2` in the linear combination, and allow correlation between random slopes `B1` and intercepts `B0`

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id] c.x2#B1[id]) ///
    covariance(B0 B1, unstructured)
```

Specify a heteroskedastic within-subject error variance that varies as a power of `x2`

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id] c.x2#B1[id]) ///
    covariance(B0 B1, unstructured) resvariance(power x2)
```

Display random-effects and within-group error parameters as standard deviations and correlations

```
menl, stddeviations
```

Three-level nonlinear regression of y on variable time and factor variable f with random intercepts S0 by lev3 and W0 by lev2 nested within lev3, using an AR(1) correlation structure for the residuals

```
menl y = {phi1:}+{phi2:}*exp(-{phi3}*time), ///
    define(phi1: i.f S0[lev3]) define(phi2: i.f W0[lev3>lev2]) ///
    rescorrelation(ar 1, t(time))
```

Three-level nonlinear regression of y on x1 with random intercepts W0 and slopes W1 on continuous x1 by lev3 and with random intercepts S0 and slopes S1 on x1 by lev2 nested within lev3, using unstructured covariance for W0 and W1 and exchangeable covariance for S0 and S1

```
menl y = {phi1:}+{b1}*cos({b2}*x1),                          ///
    define(phi1:x1 W0[lev3] S0[lev3>lev2]                    ///
                   c.x1#(W1[lev3] S1[lev3>lev2]))            ///
    covariance(W0 W1, unstructured)                          ///
    covariance(S0 S1, exchangeable)
```

As above, but assume that residuals are independent but have different variances for males and females

```
menl y = {phi1:}+{b1}*cos({b2}*x1),                          ///
    define(phi1:x1 W0[lev3] S0[lev3>lev2]                    ///
                   c.x1#(W1[lev3] S1[lev3>lev2]))            ///
    covariance(W0 W1, unstructured)                          ///
    covariance(S0 S1, exchangeable)                          ///
    rescovariance(identity, by(female))
```

Menu

Statistics > Multilevel mixed-effects models > Nonlinear regression

Syntax

```
menl depvar = <menlexpr> [if] [in] [, options]
```

<*menlexpr*> defines a nonlinear regression function as a substitutable expression that contains model parameters and random effects specified in braces {}, as in `exp({b}+{U[id]})`; see *Random-effects substitutable expressions* for details.

options	Description
Model	
`mle`	fit model via maximum likelihood; the default
`reml`	fit model via restricted maximum likelihood
`define(`*name*`:<`*resubexpr*`>)`	define a function of model parameters; this option may be repeated
`covariance(`*covspec*`)`	variance–covariance structure of the random effects; this option may be repeated
`initial(`*initial_values*`)`	initial values for parameters
Residuals	
`rescovariance(`*rescovspec*`)`	covariance structure for within-group errors
`resvariance(`*resvarspec*`)`	heteroskedastic variance structure for within-group errors
`rescorrelation(`*rescorrspec*`)`	correlation structure for within-group errors
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`variance`	show random-effects and within-group error parameter estimates as variances and covariances; the default
`stddeviations`	show random-effects and within-group error parameter estimates as standard deviations and correlations
`noretable`	suppress random-effects table
`nofetable`	suppress fixed-effects table
`estmetric`	show parameter estimates in the estimation metric
`nolegend`	suppress table expression legend
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
`nostderr`	do not estimate standard errors of random-effects parameters
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
EM options	
`emiterate(#)`	number of EM iterations; default is `emiterate(25)`
`emtolerance(#)`	EM convergence tolerance; default is `emtolerance(1e-10)`
`emlog`	show EM iteration log
Maximization	
menlmaxopts	control the maximization process
`coeflegend`	display legend instead of statistics

`coeflegend` does not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

The syntax of *covspec* is

rename1 rename2 [...] , *vartype*

vartype	Description
independent	one unique variance parameter per random effect; all covariances are 0; the default
exchangeable	equal variances for random effects and one common pairwise covariance
identity	equal variances for random effects; all covariances are 0
unstructured	all variances and covariances to be distinctly estimated

The syntax of *rescovspec* is

rescov [, *rescovopts*]

rescov	Description
identity	uncorrelated within-group errors with one common variance; the default
independent	uncorrelated within-group errors with distinct variances
ar [#]	within-group errors with autoregressive (AR) structure of order #, AR(#); ar 1 is implied by ar
ma [#]	within-group errors with moving-average (MA) structure of order #, MA(#); ma 1 is implied by ma
ctar1	within-group errors with continuous-time AR(1) structure

The syntax of *resvarspec* is

resvarfunc [, *resvaropts*]

resvarfunc	Description
identity	equal within-group error variances; the default
linear *varname*	within-group error variance varies linearly with *varname*
power *varname* | _yhat	variance function is a power of *varname* or of predicted mean
exponential *varname* | _yhat	variance function is exponential of *varname* or of predicted mean

The syntax of *rescorrspec* is

rescorr [, *rescorropts*]

rescorr	Description
identity	uncorrelated within-group errors; the default
ar [#]	within-group errors with AR(#) structure; ar 1 is implied by ar
ma [#]	within-group errors with MA(#) structure; ma 1 is implied by ma
ctar1	within-group errors with continuous-time AR(1) structure

Options

Model

`mle` and `reml` specify the statistical method for fitting the model.

`mle`, the default, specifies that the model be fit using maximum likelihood (ML).

`reml` specifies that the model be fit using restricted maximum likelihood (REML), also known as residual maximum likelihood.

`define(`*name*`:<`*resubexpr*`>)` defines a function of model parameters, <*resubexpr*>, and labels it as *name*. This option can be repeated to define multiple functions. The `define()` option is useful for expressions that appear multiple times in the main nonlinear specification *menlexpr*: you define the expression once and then simply refer to it by using `{`*name*`:}` in the nonlinear specification. This option can also be used for notational convenience. See *Random-effects substitutable expressions* for how to specify <*resubexpr*>.

`covariance(`*rename1 rename2* [...]`,` *vartype*`)` specifies the structure of the covariance matrix for the random effects. *rename1*, *rename2*, and so on, are the names of the random effects to be correlated (see *Random effects*), and *vartype* is one of the following: `independent`, `exchangeable`, `identity`, or `unstructured`. Instead of *rename*s, you can specify *restub*`*` to refer to random effects that share the same *restub* in their names.

`independent` allows for a distinct variance for each random effect and assumes that all covariances are 0; the default.

`exchangeable` specifies one common variance for all random effects and one common pairwise covariance.

`identity` is short for "multiple of the identity"; that is, all variances are equal, and all covariances are 0.

`unstructured` allows for all variances and covariances to be distinct. If p random effects are specified, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`initial(`*initial_values*`)` specifies the initial values for model parameters. You can specify a $1 \times k$ matrix, where k is the total number of parameters in the model, or you can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example, to initialize `{alpha}` to 1.23 and `{delta}` to 4.57, you would type

```
. menl ..., initial(alpha 1.23 delta 4.57) ...
```

To initialize multiple parameters that have the same group name, for example, `{y:x1}` and `{y:x2}`, with the same initial value, you can simply type

```
. menl ..., initial({y:} 1) ...
```

For the full specification, see *Specifying initial values*.

Residuals

`menl` provides two ways to model the within-group error covariance structure, sometimes also referred to as residual covariance structure in the literature. You can model the covariance directly by using the `rescovariance()` option or indirectly by using the `resvariance()` and `rescorrelation()` options.

`rescovariance(`*rescov* [`,` *rescovopts*]`)` specifies the within-group errors covariance structure or covariance structure of the residuals within the lowest-level group of the nonlinear mixed-effects model. For example, if you are modeling random effects for classes nested within schools, then

`rescovariance()` refers to the residual variance–covariance structure of the observations within classes, the lowest-level groups.

rescov is one of the following: `identity`, `independent`, `ar` [*#*], `ma` [*#*], or `ctar1`. Below, we describe each *rescov* with its specific options *rescovopts*:

`identity` [`, by(`*byvar*`)`], the default, specifies that all within-group errors be independent and identically distributed (i.i.d.) with one common error variance σ^2_ϵ. When combined with `by(`*byvar*`)`, independence is still assumed, but you estimate a distinct variance for each category of *byvar*.

`independent, index(`*varname*`)` specifies that within-group errors are independent with distinct variances for each value (index) of *varname*. `index(`*varname*`)` is required.

`ar` [*#*]`, t(`*timevar*`)` [`by(`*byvar*`)`] assumes that within-group errors have an AR(*#*) structure. If *#* is omitted, `ar 1` is assumed. `t(`*timevar*`)` is required.

`ma` [*#*]`, t(`*timevar*`)` [`by(`*byvar*`)`] assumes that within-group errors have an MA(*#*) structure. If *#* is omitted, `ma 1` is assumed. `t(`*timevar*`)` is required.

`ctar1, t(`*timevar*`)` [`by(`*byvar*`)`] assumes that within-group errors have a continuous-time AR(1) structure. This is a generalization of the AR covariance structure that allows for unequally spaced and noninteger time values. `t(`*timevar*`)` is required.

rescovopts are `index(`*varname*`)`, `t(`*timevar*`)`, and `by(`*byvar*`)`.

`index(`*varname*`)` is used within the `rescovariance()` option with *rescov* `independent`. *varname* is a nonnegative-integer–valued variable that identifies the observations within the lowest-level groups (for example, `obsid`). The groups may be unbalanced in that different groups may have different `index()` values, but you may not have repeated `index()` values within any particular group.

`t(`*timevar*`)` is used within the `rescovariance()` option to specify a time variable for the `ar`, `ma`, and `ctar1` structures.

With *rescov* `ar` and `ma`, *timevar* is an integer-valued time variable used to order the observations within the lowest-level groups and to determine the lags between successive observations. Any nonconsecutive time values will be treated as gaps. For the `ar` or `ma` structure, $\# + 1$ parameters are estimated: *#* AR or *#* MA coefficients and one overall error variance σ^2_ϵ.

With *rescov* `ctar1`, *timevar* is a real-valued time variable. The correlation between two error terms is the parameter ρ, raised to a power equal to the absolute value of the difference between the `t()` values for those errors. For the `ctar1` structure, two parameters are estimated: the correlation parameter ρ and one overall error variance σ^2_ϵ.

`by(`*byvar*`)` is for use within the `rescovariance()` option and specifies that a set of distinct within-group error covariance parameters be estimated for each category of *byvar*. In other words, you can use `by()` to model heteroskedasticity. *byvar* must be nonnegative-integer valued and constant within the lowest-level groups.

`resvariance(`*resvarfunc* [`,` *resvaropts*]`)` specifies a heteroskedastic variance structure of the within-group errors. It can be used with the `rescorrelation()` option to specify flexible within-group error covariance structures. The heteroskedastic variance structure is modeled as $\text{Var}(\epsilon_{ij}) = \sigma^2 g^2(\boldsymbol{\delta}, \upsilon_{ij})$, where σ is an unknown scale parameter, $g(\cdot)$ is a function that models heteroskedasticity (also known as variance function in the literature), $\boldsymbol{\delta}$ is a vector of unknown parameters of the variance function, and υ_{ij}'s are the values of a fixed covariate x_{ij} or of the predicted mean $\widehat{\mu}_{ij}$.

resvarfunc, omitting the arguments, is one of the following: `identity`, `linear`, `power`, or `exponential`, and *resvaropts* are options specific to each variance function.

`identity`, the default, specifies a homoskedastic variance structure for the within-group errors; $g(\boldsymbol{\delta}, v_{ij}) = 1$, so that $\text{Var}(\epsilon_{ij}) = \sigma^2 = \sigma^2_\epsilon$.

`linear` *varname* specifies that the within-group error variance vary linearly with *varname*; that is, $g(\boldsymbol{\delta}, v_{ij}) = \sqrt{varname_{ij}}$, so that $\text{Var}(\epsilon_{ij}) = \sigma^2 varname_{ij}$. *varname* must be positive.

`power` *varname*`|` `_yhat` [`, strata(`*stratavar*`)` `noconstant`] specifies that the within-group error variance, or more precisely the variance function, be expressed in terms of a power of either *varname* or the predicted mean `_yhat`, plus a constant term; $g(\boldsymbol{\delta}, v_{ij}) = |v_{ij}|^{\delta_1} + \delta_2$. If `noconstant` is specified, the constant term δ_2 is suppressed. In general, three parameters are estimated: a scale parameter σ, the power δ_1, and the constant term δ_2. When `strata(`*stratavar*`)` is specified, the power and constant parameters (but not the scale) are distinctly estimated for each stratum. A total number of $2L + 1$ parameters are estimated (L power parameters, L constant parameters, and scale σ), where L is the number of strata defined by variable *stratavar*.

`exponential` *varname*`|` `_yhat` [`, strata(`*stratavar*`)`] specifies that the within-group error variance vary exponentially with *varname* or with the predicted mean `_yhat`; $g(\gamma, v_{ij}) = \exp(\gamma v_{ij})$. Two parameters are estimated: a scale parameter σ and an exponential parameter γ. When `strata(`*stratavar*`)` is specified, the exponential parameter γ (but not scale σ) is distinctly estimated for each stratum. A total number of $L + 1$ parameters are estimated (L exponential parameters and scale σ), where L is the number of strata defined by variable *stratavar*.

resvaropts are `strata(`*stratavar*`)` and `noconstant`.

`strata(`*stratavar*`)` is used within the `resvariance()` option with *resvarfunc* `power` and `exponential`. `strata()` specifies that the parameters of the variance function $g(\cdot)$ be distinctly estimated for each stratum. The scale parameter σ remains constant across strata. In contrast, `rescovariance()`'s `by(`*byvar*`)` suboption specifies that all covariance parameters, including σ (whenever applicable), be estimated distinctly for each category of *byvar*. *stratavar* must be nonnegative-integer valued and constant within the lowest-level groups.

`noconstant` is used within the `resvariance()` option with *resvarfunc* `power`. `noconstant` specifies that the constant parameter be suppressed in the expression of the variance function $g(\cdot)$.

`rescorrelation(`*rescorr* [`,` *rescorropts*]`)` specifies a correlation structure of the within-group errors. It can be used with the `resvariance()` option to specify flexible within-group error covariance structures.

rescorr is one of the following: `identity`, `ar` [*#*], `ma` [*#*], or `ctar1`.

`identity`, the default, specifies that all within-group error correlations be zeros.

`ar` [*#*]`, t(`*timevar*`)` [`by(`*byvar*`)`] assumes that within-group errors have an AR(*#*) correlation structure. If *#* is omitted, `ar 1` is assumed. The `t(`*timevar*`)` option is required.

`ma` [*#*]`, t(`*timevar*`)` [`by(`*byvar*`)`] assumes that within-group errors have an MA(*#*) correlation structure. If *#* is omitted, `ma 1` is assumed. The `t(`*timevar*`)` option is required.

`ctar1, t(`*timevar*`)` [`by(`*byvar*`)`] assumes that within-group errors have a continuous-time AR(1) correlation structure. The `t(`*timevar*`)` option is required.

rescorropts are `t(`*timevar*`)` and `by(`*byvar*`)`.

`t(`*timevar*`)` is used within the `rescorrelation()` option to specify a time variable for the `ar`, `ma`, and `ctar1` structures.

With *rescorr* `ar` and `ma`, *timevar* is an integer-valued time variable used to order the observations within the lowest-level groups and to determine the lags between successive observations. Any nonconsecutive time values will be treated as gaps. For the `ar` or `ma` structure, # AR or MA coefficients are estimated.

With *rescorr* `ctar1`, *timevar* is a real-valued time variable. For the continuous AR(1) correlation model, the correlation between two errors is the parameter ρ, raised to a power equal to the absolute value of the difference between the `t()` values for those errors.

`by(`*byvar*`)` is used within the `rescorrelation()` option and specifies that a set of distinct within-group error correlation parameters be estimated for each category of *byvar*. *byvar* must be nonnegative-integer valued and constant within the lowest-level groups.

Reporting

`level(#)`; see [R] **estimation options**.

`variance`, the default, displays the random-effects and within-group error parameter estimates as variances and covariances.

`stddeviations` displays the random-effects and within-group error parameter estimates as standard deviations and correlations.

`noretable` suppresses the random-effects table from the output.

`nofetable` suppresses the fixed-effects table from the output.

`estmetric` displays all parameter estimates in the estimation metric. Fixed-effects estimates are unchanged from those normally displayed, but random-effects parameter estimates are displayed as log-standard deviations and hyperbolic arctangents of correlations. Within-group error parameter estimates are also displayed in their original estimation metric.

`nolegend` suppresses the expression legend that appears before the fixed-effects estimation table when functions of parameters or named substitutable expressions are specified in the main equation or in the `define()` options.

`noheader` suppresses the output header, either at estimation or upon replay.

`nogroup` suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

`nostderr` prevents `menl` from calculating standard errors for the estimated random-effects parameters, although standard errors are still provided for the fixed-effects parameters. Specifying this option will speed up computation times.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **estimation options**.

EM options

These options control the EM iterations that occur before estimation switches to the Lindstrom–Bates method. EM is used to obtain starting values.

`emiterate(#)` specifies the number of EM iterations to perform. The default is `emiterate(25)`.

`emtolerance(#)` specifies the convergence tolerance for the EM algorithm. The default is `emtolerance(1e-10)`. EM iterations will be halted once the log (restricted) likelihood changes by a relative amount less than #. At that point, optimization switches to the Lindstrom–Bates method.

`emlog` specifies that the EM iteration log be shown. The EM iteration log is not displayed by default.

Maximization

menlmaxopts: `iterate(#)`, `tolerance(#)`, `ltolerance(#)`, `pnlsopts()`, `lmeopts()`, [`no`]`log`. The convergence is declared when either `tolerance()` or `ltolerance()` is satisfied; see *Stopping rules* for details.

`iterate(#)` specifies the maximum number of iterations for the Lindstrom–Bates alternating algorithm. One iteration of the alternating algorithm involves $\#_{\text{pnls}}$ penalized least-squares (PNLS) iterations as specified in `pnlsopts()`'s `iterate()` suboption and $\#_{\text{lme}}$ linear mixed-effects (LME) iterations as specified in `lmeopts()`'s `iterate()` suboption. The default is the current value of `set maxiter`, which is `iterate(16000)` by default.

`tolerance(#)` specifies the tolerance for the parameter vector. When the relative change in the parameter vector from one alternating algorithm iteration to the next is less than or equal to `tolerance()`, the parameter convergence is satisfied. The default is `tolerance(1e-6)`.

`ltolerance(#)` specifies the tolerance for the linearization log likelihood in the Lindstrom–Bates alternating algorithm. The linearization log likelihood is the log likelihood from the LME optimization step in the last iteration. When the relative change in the linearization log likelihood from one alternating algorithm iteration to the next is less than or equal to `ltolerance()`, the log-likelihood convergence is satisfied. The default is `ltolerance(1e-7)`.

`pnlsopts(`*pnlsopts*`)` controls the PNLS optimization step of the Lindstrom–Bates alternating algorithm. *pnlsopts* include any of the following: `iterate(#)`, `ltolerance(#)`, `tolerance(#)`, `nrtolerance(#)`, and *maximize_options*. The convergence of this step within each alternating iteration is declared when `nrtolerance()` and one of `tolerance()` or `ltolerance()` are satisfied.

`iterate(#)` specifies the maximum number of iterations for the PNLS optimization step within the alternating algorithm. The default is `iterate(5)`.

`ltolerance(#)` specifies the tolerance for the objective function in the PNLS optimization step. When the relative change in the objective function from one PNLS iteration to the next is less than or equal to `ltolerance()`, the objective-function convergence is satisfied. The default is `ltolerance(1e-7)`.

`tolerance(#)` specifies the tolerance for the vector of fixed-effects parameters. When the relative change in the coefficient vector from one PNLS iteration to the next is less than or equal to `tolerance()`, the parameter convergence criterion is satisfied. The default is `tolerance(1e-6)`.

`nrtolerance(#)` specifies the tolerance for the scaled gradient in the PNLS optimization step. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

maximize_options are [`no`]`log`, `trace`, `showtolerance`, `nonrtolerance`; see [R] **maximize**.

`lmeopts(`*lmeopts*`)` controls the LME optimization step of the Lindstrom–Bates alternating algorithm. *lmeopts* include any of the following: `iterate(#)`, `ltolerance(#)`, `tolerance(#)`, `nrtolerance(#)`, and *maximize_options*. The convergence of this step within each alternating iteration is declared when `nrtolerance()` and one of `tolerance()` or `ltolerance()` are satisfied.

`iterate(#)` specifies the maximum number of iterations for the LME optimization step within the alternating algorithm. The default is `iterate(5)`.

`ltolerance(#)` specifies the tolerance for the log likelihood in the LME optimization step. When the relative change in the log likelihood from one LME iteration to the next is less than or equal to `ltolerance()`, the log-likelihood convergence is satisfied. The default is `ltolerance(1e-7)`.

`tolerance(#)` specifies the tolerance for the random-effects and within-group error covariance parameters. When the relative change in the vector of parameters from one LME iteration to the next is less than or equal to `tolerance()`, the convergence criterion for covariance parameters is satisfied. The default is `tolerance(1e-6)`.

`nrtolerance(#)` specifies the tolerance for the scaled gradient in the LME optimization step. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

maximize_options are [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `nonrtolerance`; see [R] **maximize**.

[`no`]`log`; see [R] **maximize**.

The following option is available with `menl` but is not shown in the dialog box:

`coeflegend`; see [R] **estimation options**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction
Random-effects substitutable expressions
Substitutable expressions
Linear combinations
Linear forms versus linear combinations
Random effects
Multilevel specifications
Summary
Specifying initial values
Two-level models
Testing variance components
Random-effects covariance structures
Heteroskedastic within-group errors
Restricted maximum likelihood
Three-level models
Obtaining initial values
Linearization approach to finding initial values
Graphical approach to finding initial values
Smart regressions approach to finding initial values
Examples of specifying initial values

Introduction

Nonlinear mixed-effects (NLME) models are models containing both fixed effects and random effects where some of, or all, the fixed and random effects enter the model nonlinearly. They can be viewed as a generalization of linear mixed-effects (LME) models (see [ME] **mixed**), in which the conditional mean of the outcome given the random effects is a nonlinear function of the coefficients and random effects. Alternatively, they can be considered as an extension of nonlinear regression models for independent data (see [R] **nl**), in which coefficients may incorporate random effects, allowing them to vary across different levels of hierarchy and thus inducing correlation within observations at the same level.

Why use NLME models? Can't we use higher-order polynomial LME models or generalized linear mixed-effects (GLME) models instead?

In principle, any smooth nonlinear function can be approximated by a higher-order polynomial. One may argue that we can use an LME (see [ME] **mixed**) polynomial model and increase the order of the polynomial until we get an accurate approximation of the desired nonlinear model. There are three problems with this approach. First, parameters in NLME models often have natural physical interpretations such as half-life and limiting growth. This is not the case in LME polynomial models. For example, what is the physical interpretation of the coefficient of `time`^{4}? Second, NLME models typically use fewer parameters than the corresponding LME polynomial model, which provides a more parsimonious summarization of the data. Third, NLME models usually provide better predictions outside the range of the observed data than predictions based on LME higher-order polynomial models.

GLME models (see [ME] **meglm**) are also nonlinear, but in the restricted sense that the conditional mean response given random effects is a nonlinear function of the linear predictor that contains both fixed and random effects, and only indirectly nonlinear in fixed and random effects themselves. That is, the nonlinear function must be an invertible function of the linear predictor. However, many estimation methods for GLME and NLME models are similar because random effects enter both models nonlinearly.

Population pharmacokinetics, bioassays, and studies of biological and agricultural growth processes are just a few areas that use NLME models to analyze multilevel data such as longitudinal or repeated-measures data. Comprehensive treatments of both methodology and history of NLME models may be found in Davidian and Giltinan (1995), Vonesh and Chinchilli (1997), Demidenko (2013), and Pinheiro and Bates (2000). Davidian and Giltinan (2003) provide an excellent summary.

Consider a sample of M subjects from a population of interest, where n_j measurements, $y_{1j}, \ldots, y_{n_j j}$, are observed on subject j at times $t_{1j}, \ldots, t_{n_j j}$. By "subject", we mean any distinct experimental unit, individual, panel, or cluster with two or more correlated observations. The basic nonlinear two-level model can be written as follows (in our terminology, a one-level NLME is just a nonlinear regression model for independent data),

$$y_{ij} = \mu\left(\mathbf{x}'_{ij}, \boldsymbol{\beta}, \mathbf{u}_j\right) + \epsilon_{ij} \qquad i = 1, \ldots, n_j;\ j = 1, \ldots, M \tag{1}$$

where $\mu(\cdot)$ is a real-valued function that depends on a $p \times 1$ vector of fixed effects $\boldsymbol{\beta}$, a $q \times 1$ vector of random effects $\mathbf{u}_j$, which are distributed as multivariate normal with mean $\mathbf{0}$ and variance–covariance matrix $\boldsymbol{\Sigma}$, and a covariate vector $\mathbf{x}_{ij}$ that contains both within-subject covariates $\mathbf{x}^w_{ij}$ and between-subject covariates $\mathbf{x}^b_j$. The $n_j \times 1$ vector of errors $\boldsymbol{\epsilon}_j = \left(\epsilon_{1j}, \ldots, \epsilon_{n_j j}\right)'$ is assumed to be multivariate normal with mean $\mathbf{0}$ and variance–covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$, where depending on $\boldsymbol{\Lambda}_j$, σ^2 is either a within-group error variance σ^2_ϵ or a squared scale parameter σ^2.

Parameters of NLME models often have scientifically meaningful interpretations, and research questions are formed based on them. To allow parameters to reflect phenomena of interest, (1) can be equivalently formulated as a two-stage hierarchical model as follows:

$$\begin{aligned} \text{Stage 1: Individual-level model } y_{ij} &= m\left(\mathbf{x}_{ij}^{w},\, \boldsymbol{\phi}_j\right) + \epsilon_{ij} \qquad i = 1, \ldots, n_j \\ \text{Stage 2: Group-level model } \boldsymbol{\phi}_j &= \boldsymbol{d}\left(\mathbf{x}_j^{b},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) \qquad j = 1, \ldots, M \end{aligned} \tag{2}$$

In stage 1, we model the response by using a function $m(\cdot)$, which describes within-subject behavior. This function depends on subject-specific parameters $\boldsymbol{\phi}_j$'s, which have a natural physical interpretation, and a vector of within-subject covariates $\mathbf{x}_{ij}^{w}$. In stage 2, we use a known vector-valued function $\boldsymbol{d}(\cdot)$ to model between-subject behavior, that is, to model $\boldsymbol{\phi}_j$'s and to explain how they vary across subjects. The $\boldsymbol{d}(\cdot)$ function incorporates random effects and, optionally, a vector of between-subject covariates $\mathbf{x}_j^{b}$. The general idea is to specify a common functional form for each subject in stage 1 and then allow some parameters to vary randomly across subjects in stage 2.

To further illustrate (1) and (2), we consider the soybean plants data (Davidian and Giltinan 1995), in which we model the average leaf weight per soybean plant, y_{ij}, in plot j at t_{ij} days after planting. Let's first use (1):

$$\begin{aligned} y_{ij} &= \mu\left(\mathbf{x}_{ij}',\, \boldsymbol{\beta},\, \mathbf{u}_j\right) + \epsilon_{ij} \\ &= \frac{\beta_1 + u_{1j}}{1 + \exp\left[-\left\{t_{ij} - (\beta_2 + u_{2j})\right\} / (\beta_3 + u_{3j})\right]} + \epsilon_{ij} \end{aligned}$$

Here $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3)'$, $\mathbf{u}_j = (u_{1j}, u_{2j}, u_{3j})'$, and $\mathbf{x}_{ij}$ is simply t_{ij}.

Equivalently, we can use (2) to define our model,

$$\begin{aligned} \text{Stage 1: } y_{ij} &= m\left(\mathbf{x}_{ij}^{w},\, \boldsymbol{\phi}_j\right) + \epsilon_{ij} \\ &= \frac{\phi_{1j}}{1 + \exp\left\{-\left(t_{ij} - \phi_{2j}\right) / \phi_{3j}\right\}} + \epsilon_{ij} \\ \text{Stage 2: } \phi_{1j} &= \beta_1 + u_{1j} \\ \phi_{2j} &= \beta_2 + u_{2j} \\ \phi_{3j} &= \beta_3 + u_{3j} \end{aligned}$$

where $\mathbf{x}_{ij}^{w} = t_{ij}$, $\boldsymbol{\phi}_j = (\phi_{1j}, \phi_{2j}, \phi_{3j})' = \boldsymbol{d}\left(\mathbf{x}_j^{b},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) = \boldsymbol{\beta} + \mathbf{u}_j$. A key advantage of (2) is the interpretability. $\boldsymbol{\phi}_j$'s are parameters that characterize features of the trajectory. For example, ϕ_{1j} can be interpreted as the asymptotic average leaf weight per soybean plant in plot j when $t_{ij} \to \infty$ and ϕ_{2j} as the time at which half of ϕ_{1j} is reached; that is, if we set $t_{ij} = \phi_{2j}$, then $E(y_{ij}) = \phi_{1j}/2$. `menl` provides both representations.

The random effects $\mathbf{u}_j$ are not directly estimated (although they may be predicted) but instead are characterized by the elements of $\boldsymbol{\Sigma}$, known as variance components, which are estimated together with the parameters of the within-group error variance–covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$. Correlation among repeated measures is induced either indirectly through the subject-specific random effects $\mathbf{u}_j$ or directly through specification of the within-subject covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$. Several covariance structures are available for $\boldsymbol{\Sigma}$, similar to those allowed in `mixed`. In contrast to `mixed`, `menl` provides more flexible modeling of the within-subject variance and correlation structures.

menl uses the following decomposition of the $\mathbf{\Lambda}_j$ matrix,

$$\mathbf{\Lambda}_j = \mathbf{S}_j \mathbf{C}_j \mathbf{S}_j \tag{3}$$

where $\mathbf{S}_j$ is diagonal with positive elements such that $\text{Var}(\epsilon_{ij}) = \sigma^2 [\mathbf{S}_j]^2_{ii}$ and $\mathbf{C}_j$ is a correlation matrix such that $\text{corr}(\epsilon_{ij}, \epsilon_{kj}) = [\mathbf{C}_j]_{ik}$; $[A]_{ij}$ denotes the ijth element of matrix A. Decomposition (3) of $\mathbf{\Lambda}_j$ allows us to separately model the variance structure (heteroskedasticity) and the correlation structure by using disjoint sets of parameters for $\mathbf{C}_j$ and $\mathbf{S}_j$. This is different from how mixed handles within-subject correlation, where heteroskedasticity and correlation are determined by the type of the chosen residual covariance structure. For convenience, menl accommodates the behavior of the mixed command for specifying residual covariance structures via the rescovariance() option. The more flexible modeling of the residual structures according to (3) is available via the resvariance() and rescorrelation() options.

For LME models, because the random effects $\mathbf{u}_j$'s are unobserved, inference about $\boldsymbol{\beta}$ and the covariance parameters are based on the marginal likelihood obtained after integrating out the random effects. Unlike LME models, no closed-form solution is available because the random effects enter the model nonlinearly, making the integration analytically intractable in all but the simplest situations. There are two principal methods proposed in the literature for fitting NLME models. One is to use an adaptive Gauss–Hermite (AGH) quadrature to approximate the integral that appears in the expression of the marginal likelihood. The other one is to use the linearization method of Lindstrom and Bates (1990), also known as a conditional first-order linearization method, which is based on a first-order Taylor-series approximation of the mean function and essentially linearizes the mean function with respect to fixed and random effects. With the AGH method, the level of accuracy increases as the number of quadrature points increases but at the expense of increasing computational burden. The linearization method is computationally efficient because it avoids the intractable integration, but the approximation cannot be made arbitrarily accurate. Despite its potential limiting accuracy, the linearization method has proven the most popular in practice (Fitzmaurice et al. 2009, sec. 5.4.8). The linearization method of Lindstrom and Bates (1990), with extensions from Pinheiro and Bates (1995), is the method of estimation in menl.

The linearization method uses a first-order Taylor-series expansion of the specified nonlinear mean function to approximate it with a linear function of fixed and random effects. Thus an NLME model is approximated by an LME model, in which the fixed-effects and random-effects design matrices involve derivatives of the nonlinear mean function with respect to fixed effects (coefficients) and random effects, respectively. As such, inference after the linearization method uses the computational machinery of the LME models. For example, estimates of random effects are computed as best linear unbiased predictors (BLUPs) of random effects from the approximating LME model. The accuracy of the inferential results will depend on the accuracy of the linearization method in approximating the original NLME model. In general, asymptotic inference for the NLME models based on the linearization method is only "approximately asymptotic", making it less accurate than the corresponding asymptotic inference for true LME models. In practice, however, the linearization method was found to perform well in many situations (for example, Pinheiro and Bates [1995]; Wolfinger and Lin [1997]; Plan et al. [2012]; and Harring and Liu [2016]).

Both ML and REML estimation are supported by menl. The ML estimates are based on the usual application of likelihood theory, given the distributional assumptions of the model. In small samples, ML estimation generally leads to small-sample bias in the estimated variance components. The REML method (Thompson 1962) reduces this bias by forming a set of linear contrasts of the response that do not depend on the fixed effects $\boldsymbol{\beta}$ but instead depend only on the variance components to be estimated. The likelihood is then formed based on the distribution of the linear contrasts, and standard ML methods are applied.

The next section describes how to specify nonlinear expressions containing random effects in menl.

Random-effects substitutable expressions

You define the nonlinear model to be fit by `menl` by using a random-effects substitutable expression, a substitutable expression that contains random effects. For example, `exp({b}+{U[id]})`, `{b1}/({b2}+{b3}*x+{U[id]})`, and `({b1}+{U1[id]})/(1+{b2}*x+{c.x#U2[id]})` are a few examples of such expressions. We describe them in more detail below.

Substitutable expressions

Let's first consider substitutable expressions without random effects. Substitutable expressions are just like any other mathematical expressions involving scalars and variables, such as those you would use with Stata's `generate` command, except that the parameters to be estimated are bound in braces. See **[U] 13.2 Operators** and **[U] 13.3 Functions** for more information on expressions.

Although `menl` requires that you include random effects in your model, for teaching purposes, we will start with simpler substitutable expressions that do not contain random effects. Suppose that we wish to fit the model

$$y_{ij} = \alpha\left(1 - e^{-(\beta_0+\beta_1 x_{1ij}+\beta_2 x_{2ij})}\right) + \epsilon_{ij}$$

where α, β_0, β_1, and β_2 are the parameters to be estimated and ϵ_{ij} is an error term. We could simply type

```
. menl y = {a}*(1 - exp(-({b0}+{b1}*x1+{b2}*x2)))
```

Because `a`, `b0`, `b1`, and `b2` are enclosed in braces, `menl` knows that they are parameters in the model.

You can group several parameters together by assigning a group name (or equation name) to them. Parameters with the same group name, `lc` in the example below, will be grouped together in the output table:

```
. menl y = {a}*(1 - exp(-({lc:b0}+{lc:b1}*x1+{lc:b2}*x2)))
```

That is, parameters `b0`, `b1`, and `b2` will appear together in the output table in the equation labeled `lc`. Parameters without equation names will appear at the bottom of the output table.

Sometimes, it may be convenient to define subexpressions within the main expression. This can be done inside the expression itself or by using the `define()` option. For example,

```
. menl y = {a}*(1 - exp(-{xb:})), define(xb: {lc:b0}+{lc:b1}*x1+{lc:b2}*x2)
```

defines the linear predictor of the exponent in the `define()` option with label `xb` and then refers to it inside the exponent as `{xb:}`. You can define as many subexpressions as you like by using the `define()` option repeatedly. Defining subexpressions is also useful for later predictions; see, for instance, example 13.

The above is equivalent to

```
. menl y = {a}*(1 - exp(-{xb: {lc:b0}+{lc:b1}*x1+{lc:b2}*x2}))
```

Parameters `{a}`, `{lc:b0}`, `{lc:b1}`, and `{lc:b2}` are what we call "free parameters", meaning that they are not defined by a linear form, which we describe in the next section. Free parameters are displayed with a forward slash in front of their names or their group names.

The general syntax for a free parameter is

{[*eqname*:] *name*}

Linear combinations

Nonlinear functions will often contain linear combinations of variables. Recall our nonlinear function from *Substitutable expressions*:

$$y_{ij} = \alpha\left(1 - e^{-(\beta_0+\beta_1 x_{1ij}+\beta_2 x_{2ij})}\right) + \epsilon_{ij}$$

Instead of explicitly specifying the linear combination that appears in the exponent, as we did in the previous section, we can use `menl`'s shorthand notation

```
. menl y = {a}*(1 - exp(-({lc: x1 x2})))
```

By specifying `{lc:x1 x2}`, you are telling `menl` that you are declaring a linear combination named `lc` that is a function of two variables, `x1` and `x2`. `menl` will create three parameters, named `{lc:_cons}`, `{lc:x1}`, and `{lc:x2}`.

Although both specifications produce the same results, the shorthand specification is more convenient.

The general syntax for defining a linear combination is

{ *eqname*: *varspec* [, xb noconstant] }

where *varspec* includes a list of variables (*varlist*), a list of *random-effects terms*, or both.

The `xb` option is used to distinguish between the linear combination that contains one variable and a free parameter that has the same name as the variable and the same group name as the linear combination. For example, `{lc: x1, xb}` is equivalent to `{lc:_cons} + {lc:x1}*x1`, whereas `{lc:x1}` refers to either a free parameter `x1` with a group name `lc` or the coefficient of the `x1` variable, if `{lc:}` has been previously defined in the expression as a linear combination that involves variable `x1`; see examples below. Thus the `xb` option indicates that the specification is a linear combination rather than a single parameter to be estimated.

When you define a linear combination, a constant term is included by default (a mathematician would argue that "affine combination" is the correct terminology!). The `noconstant` option suppresses the constant.

Having defined a linear combination such as `{lc:x1 x2}`, you can refer to its individual coefficients by using `{lc:x1}` and `{lc:x2}` or, more generally, `{`*eqname*`:`*varname*`}`. For example, suppose that we want to fit the following model, where the coefficient of `x1`, β_1, appears in two places in the expression:

$$y_{ij} = \frac{1}{(1+\beta_1 x_{1ij}+\beta_2 x_{2ij}+\beta_3 x_{3ij})} \exp\left\{-\left(\alpha_0+\alpha_1 z_{ij}\right)/\left(1+\beta_1 x_{4ij}\right)\right\} + \epsilon_{ij}$$

We use `{lc1: x1 x2 x3, noconstant}` to specify the first linear combination, which appears in the denominator outside the exponentiated expression, and then use `{lc1:x1}` to refer to β_1 in the denominator inside the exponentiated expression. We also use the `xb` option, when we specify the second linear combination that contains only one covariate `z`. Below is the full specification:

```
. menl y = 1/(1+{lc1: x1 x2 x3, noconstant})*exp(-{lc2: z, xb}/(1+{lc1:x1}*x4))
```

You may also refer to a "subset" of a previously defined linear combination. For example, let's modify our previous expression by substituting $\beta_1 x_{4ij}$ in the denominator in the exponent with the subset $\beta_1 x_{1ij} + \beta_3 x_{3ij}$ of the first linear combination:

$$y_{ij} = \frac{1}{(1+\beta_1 x_{1ij}+\beta_2 x_{2ij}+\beta_3 x_{3ij})} \exp\left\{-\left(\alpha_0+\alpha_1 z_{ij}\right)/\left(1+\beta_1 x_{1ij}+\beta_3 x_{3ij}\right)\right\} + \epsilon_{ij}$$

The coefficients for variables x1 and x3 are the same in the denominators inside and outside the exponent. We fit this model by typing

```
. menl y = 1/(1+{lc1: x1 x2 x3, nocons})*     ///
          exp(-{lc2: z, xb}/(1+{lc1: x1 x3, nocons}))
```

We used the same equation name, lc1, to constrain the coefficients to be the same between the two linear-combination specifications. If we used a different equation name, say, lc3, in the last linear combination, we would have specified $\beta_4 x_{1ij} + \beta_5 x_{3ij}$ instead of $\beta_1 x_{1ij} + \beta_3 x_{3ij}$ and estimated two extra parameters, β_4 named {lc3:x1} and β_5 named {lc3:x3}.

To refer to the entire linear combination that was already defined, you can simply refer to its name. For example, if both denominators included the same linear combination, $\beta_1 x_{1ij} + \beta_2 x_{2ij} + \beta_3 x_{3ij}$, the corresponding menl specification would be

```
. menl y = 1/(1+{lc1: x1 x2 x3, nocons})*exp(-{lc2: z, xb}/(1+{lc1:}))
```

Just like subexpressions, linear combinations can be defined in the define() option. For example, the above is equivalent to

```
. menl y = 1/(1+{lc1:})*exp(-{lc2:}/(1+{lc1:})), define(lc1: x1 x2 x3, nocons) ///
                                                 define(lc2: z, xb)
```

Linear forms versus linear combinations

As we mentioned in *Linear combinations*, the linear-combination specification is syntactically convenient. It can also be more computationally efficient when a linear combination is a linear form.

A linear combination is what we call a linear form as long as you do not refer to its coefficients or any subset of the linear combination anywhere in the expression. Linear forms are beneficial for some nonlinear commands such as nl because they make derivative computation faster and more accurate. Although menl does not fully utilize the linear-form specification in its computations, it is still important to learn to distinguish between linear forms and linear combinations.

For example, in *Linear combinations*, the first linear combination {lc:}, the linear combination {lc2:}, and the linear combination {lc1:} in the last example are all linear forms. The linear combination {lc1:} in the examples where we referred to {lc1:x1} and {lc1:x1 x3} is not a linear form.

In contrast to free parameters, parameters of a linear form are displayed without forward slashes in the output. Rather, they are displayed as parameters within an equation whose name is the linear combination name. Parameters of linear combinations that are not linear forms are considered free parameters.

Random effects

So far, we have restricted our discussion to substitutable expressions that do not contain random effects. Examples of random effects specified within the menl syntax are {U1[id]}, {U2[id1>id2]}, {c.x1#U3[id]}, and {2.f1#U4[id]}. These represent a random intercept at the id level, a random intercept at the id2-within-id1 level, a random slope for the continuous variable x1, and a random slope associated with the second level of the factor variable f1, respectively.

The general syntax for specifying random effects, *respec*, is provided below.

respec	Description
{*rename*[*levelspec*]}	Random intercepts *rename* at hierarchy *levelspec*
{c.*varname*#*rename*[*levelspec*]}	Random coefficients *rename* for continuous variable *varname*
{#.*fvvarname*#*rename*[*levelspec*]}	Random coefficients *rename* for the #th level of factor variable *fvvarname*

rename is a random-effects name. It is a Stata name that starts with a capital letter. *levelspec* defines the level of hierarchy and is described below.

levelspec	Description
levelvar	variable identifying the group structure for the random effect at that level
lv2 > *lv1*	two-level nesting: levels of variable *lv1* are nested within *lv2*
lv3 > *lv2* > *lv1*	three-level nesting: levels of variable *lv1* are nested within *lv2*, which is nested within *lv3*
. . . > *lv3* > *lv2* > *lv1*	higher-level nesting

You can equivalently specify levels in the opposite order, from the lowest level to the highest; for example, *lv1* < *lv2* < *lv3*, but they will be displayed in the canonical order, from the highest level to the lowest.

Random effects can be specified within a linear-combination specification such as `{lc_u: x1 x2 U1[id1] U2[id2>id1]}`. In this case, the curly braces around each random effect are not needed.

Let us illustrate several random-effects specifications with `menl`. In this section, we concentrate on two-level nonlinear models; see *Multilevel specifications* for higher-level models.

Suppose that we want to fit the following model:

$$y_{ij} = \frac{\alpha z_{ij} + u_{0j}}{1 + \exp\{-(\beta_0 + \beta_1 x_{1ij})\}} + \epsilon_{ij}$$

Compared with models we considered in previous sections, this model includes random effects or, specifically, random intercepts u_{0j}. Suppose that these random intercepts correspond to the levels of the `id` variable. Then, we can include them in our model by using `{U0[id]}`, where `U0` will be their name.

```
. menl y = ({a}*z+{U0[id]})/(1+exp(-({b0}+{b1}*x1)))
```

A more efficient specification is to use the linear-combination notation:

```
. menl y = {lc1: z U0[id], nocons}/(1+exp(-{lc2: x1, xb}))
```

The curly braces around `U0[id]` are removed when it is specified within a linear-combination specification.

If you need to refer to the random-effects term again in the expression, you can simply use its name. For example, suppose that our model includes the same random intercepts in both the numerator and the denominator.

$$y_{ij} = \frac{\alpha z_{ij} + u_{0j}}{1 + \exp\{-(\beta_0 + \beta_1 x_{1ij} + u_{0j})\}} + \epsilon_{ij}$$

We include random intercepts u_{0j}'s in the second linear combination by simply referring to their name, `U0`:

```
. menl y = {lc1: z U0[id], nocons}/(1+exp(-{lc2: x1 U0}))
```

If instead of u_{0j}'s, we had a different set of random intercepts, v_{0j}'s, in the denominator, we would need to specify a new set of random intercepts, say, `V0[id]`, with `menl`:

```
. menl y = {lc1: z U0[id], nocons}/(1+exp(-{lc2: x1 V0[id]}))
```

The shorthand notation for referring to random effects only by name, that is, without the brackets and the *levelspec*, is also useful when specifying the `covariance()` option, especially for multilevel random effects with long-level specifications; see *Multilevel specifications*.

Let's now see how to include random slopes. Consider the following extension of the *first*, simpler model in this subsection:

$$y_{ij} = \frac{\alpha z_{ij} + u_{0j} + u_{1j} z_{ij}}{1 + \exp\{-(\beta_0 + \beta_1 x_{1ij})\}} + \epsilon_{ij}$$

Here u_{1j} is a random slope for a continuous variable `z` and is specified as `{c.z#U1[id]}` directly or without curly braces within a linear-combination specification.

```
. menl y = {lc1: z U0[id] c.z#U1[id], nocons}/(1+exp(-{lc2: x1, xb}))
```

We can also include random slopes for factor variables. To demonstrate this, let's consider a different nonlinear model for variety. Consider the model below, where binary variables x_{1ij} and x_{2ij} correspond to the factor levels 1 and 2 of a factor variable x that takes on values 0, 1, and 2, with 0 being the base level.

$$y_{ij} = \alpha_0 + \alpha_1 z_{1ij} - \sqrt{w_{ij}^2 + \exp\left(\beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + u_{0j} + u_{1j} x_{1ij} + u_{2j} x_{2ij}\right)} + \epsilon_{ij}$$

There are three random-effects terms in this model: random intercepts u_{0j}, random slopes u_{1j} for x_{1ij} (level 1 of x), and random slopes u_{2j} for x_{2ij} (level 2 of x). In Stata, for a factor variable `x`, we can use the factor-variable notation ([U] **11.4.3 Factor variables**) to refer to its levels, `1.x` for level 1 and `2.x` for level 2. So, to include the three random-effects terms in `menl`, we will use `U0[id]`, `1.x#U1[id]`, and `2.x#U2[id]`, respectively.

```
. menl y = {lc1: z1, xb} - sqrt(c.w#c.w +  ///
      exp({lc2: i.x U0[id] 1.x#U1[id] 2.x#U2[id]}))
```

In the above specification, we used the factor-variable notations `i.x` to include fixed effects for all levels of `x`, except the base level, and `c.w#c.w` to include a square of `w`; see [U] **11.4.3 Factor variables** for details. The factor-variable specification `i.` or any other specification that refers to multiple levels of a factor variable is not allowed when specifying random coefficients, because each level will typically require a different set of random effects. For example, if we had specified `i.x#U[id]` in the above example, we would have received an error.

Multilevel specifications

In *Random effects*, we focused on specifying substitutable expressions containing random effects for two-level nonlinear mixed-effects models. Here we will consider higher-level models.

Suppose that we want to fit the following three-level nonlinear mixed-effects model,

$$y_{ijk} = \beta_0 + u_{0k}^{(3)} + u_{0jk}^{(2)} + \cos\left\{\left(\beta_1 + u_{1k}^{(3)}\right) x_{1ijk}\right\} + \epsilon_{ijk}$$

where first-level observations, indexed by i, are nested within second-level groups, indexed by j, which are nested within third-level groups, indexed by k.

There are three random-effects terms in this model: random intercepts, $u_{0k}^{(3)}$, and random slopes for x_1, $u_{1k}^{(3)}$, at the third level (`idk`) and random intercepts $u_{0jk}^{(2)}$ at the second level (`idj`-nested-within-`idk`). We specify random intercepts and random slopes for x_1 at the highest hierarchical level just like we did in *Random effects* for two-level models. Specifically, we can use `U0[idk]` and `c.x1#U1[idk]`, respectively. To specify random intercepts $u_{0jk}^{(2)}$ at the `idj`-nested-within-`idk` level, we need to use one of the *levelspec* specifications for two nested levels. For example, we can use `UU0[idk>idj]`. Below is the full specification:

```
. menl y = {lc1: U0[idk] UU0[idk>idj]} + cos({lc2: x1 c.x1#U1[idk], noconstant})
```

We can also include a random slope of the `x1` variable at the `idj`-within-`idk` level in the cosine function by specifying `c.x1#UU1[idk>idj]` inside the `cos()` function.

```
. menl y = {lc1: U0[idk] UU0[idk>idj]} +   ///
      cos({lc2: x1 c.x1#U1[idk] c.x1#UU1[idk>idj], noconstant})
```

We can shorten the above specification by writing `c.x1#U1[idk] c.x1#UU1[idk>idj]` more compactly as `c.x1#(U1[idk] UU1[idk>idj])`,

```
. menl y = {lc1: U0[idk] UU0[idk>idj]} +   ///
      cos({lc2: x1 c.x1#(U1[idk] UU1[idk>idj]), noconstant})
```

Similarly, if we had a four-level model with, say, a random intercept at the `idj`-within-`idk`-within-`idl` level, we could specify it as `W[idl>idk>idj]`; see *levelspec* for other specifications.

Summary

To summarize, here are a few rules to keep in mind when defining substitutable expressions.

1. Model parameters and random effects are bound in braces if specified directly in the expression: `{b0}`, `{U0[id]}`, etc.
2. Model parameters can be assigned group names: `{slopes:x1}`, `{slopes:x2}`, etc.
3. Random-effects names must start with a capital letter as in `{U0[id]}`, `{c.x#U1[id]}`, `{V0[id2>id1]}`, `{1.z#V1[id2>id1]}`, etc.
4. The factor-variable specification `i.`, as in `{i.z#V1[id2>id1]}`, or any other specification that refers to multiple levels of a factor variable, as in `{i(1/4).z#V1[id2>id1]}`, is not allowed when specifying random coefficients.
5. Linear combinations of variables can be included using the specification

   {*eqname*`:`*varlist*[`, xb noconstant`]`}`

   For example, `{price: mpg weight i.rep78}` and `{lc: x1 x2, noconstant}`.
6. Random effects can be specified within a linear combination, in which case they should be included without curly braces, for example, `{lc_u: x1 x2 U[id]}`.
7. To specify a linear combination that contains only one variable, use the `xb` option, for example, `{lc: x1, xb}`.
8. To refer to the previously defined linear combination again in the expression, simply use its name {*eqname*`:`}, for example, `{lc:}` and `{lc_u:}`.
9. You can refer to individual parameters of the linear combination by using {*eqname*`:_cons`} and {*eqname*`:`*varname*}, for example, `{price:_cons}` and `{price:weight}`.

10. You can refer to a "subset" of the previously defined linear combination by using {*eqname*:*subset*}, where *subset* is a subset of the variables from *varlist* used to define *eqname*, as in {price: mpg weight}. To refer to the subset containing only one variable, use the xb option, as in {price: weight, xb}. If a linear combination contains only one random-effects term, the xb option is implied.
11. To refer to the previously defined random effects again in the expression or in the covariance() option, simply use their names, such as {U0} and {U1}.
12. You can define subexpressions, including linear combinations, inside the main expression or in the define() option, which can be repeated. For example,

```
. menl y = {numer:}/{denom:}, define(numer: z U0[id]) ///
      define(denom:1+exp(-{lc: x1, xb}))
```

13. Specify linear forms whenever possible for faster and more accurate computation of derivatives; see *Linear forms versus linear combinations*.
14. Model parameters that are not defined by linear forms are considered free parameters. They are included in the output with a forward slash in front of their names or group names and displayed after linear forms in the estimation table.

Specifying initial values

By default, menl uses the EM algorithm to obtain initial values, but you may often need to specify your own. You specify your own initial values in the initial() option. For example, specifying the initial(a 1.1 b -2) option with menl initializes parameter {a} to 1.1 and parameter {b} to −2.

When you specify your own initial values, they are used for initialization, and the EM algorithm is not performed. When you specify initial values for only a subset of model parameters, the remaining parameters are initialized with some predetermined values such as zeros for fixed-effects parameters and correlations and ones for variances. You can specify the iterate(0) option to see the initial values that will be used by menl in the optimization.

Often, you may have good initial values for fixed-effects parameters but not for random-effects parameters. In this case, you can specify initial()'s fixed suboption to supply your own fixed-effects parameters, but use the EM algorithm to obtain initial values for the random-effects parameters.

There are three ways in which you can use the initial(*initial_values*) option: you can specify a vector of values, a list of values, or values for individual parameters and groups of parameters.

Specifically, *initial_values* is one of the following:

vectorname [, skip copy fixed]

[#] [...], copy

paramlist [=] # [*paramlist* [=] # [...]] [, fixed]

skip specifies that any parameters found in the specified initialization vector, *vectorname*, that are not also found in the model be ignored. The default action is to issue an error message.

copy specifies that the initial values be copied into the initialization vector without checking for valid column names. copy must be specified when initial values are supplied as a list of numbers.

fixed specifies that initial estimates are being supplied for the fixed effects only and that menl should still perform the EM algorithm to refine initial values for variance components. The specified initial values are used for fixed-effects parameters during the EM algorithm. If you omit fixed, menl presumes that you are specifying starting values for all parameters in e(b), and the EM algorithm will not be performed.

Examples of *paramlist* are `param`, `{param}`, `{param1} {param2}`, `{param1 param2}`, `{grp:param1} {grp:param2} {grp:param3}`, `{grp:param1 param2}`, and `{grp:}`.

Let's describe each specification in more detail. You can specify the name of a vector containing the initial values, say, `initial(b0)`. Vector `b0` should be properly labeled with labels found in column names of `e(b)`, unless you specify the `copy` option. A properly labeled vector can have fewer elements than `e(b)` or, if `skip` is specified, even more elements. A vector without labels must be of the same dimension as `e(b)`.

Alternatively, you can supply a list of numbers to `initial()`, in which case `copy` must be specified. The list of numbers should be of length equal to the dimension of `e(b)`. For example, if `e(b)` has four parameters and you type `initial(1.1 0 3 -2, copy)`, then the four coefficients in `e(b)` will be initialized to 1.1, 0, 3, and −2, respectively. If instead you specify, for example, only three initial values in your list, an error will be issued.

Finally, you can initialize parameters by referring to their names. You can specify a parameter name, its initial value, another parameter name, its initial value, and so on, for example, `initial(a 1.1 b -2)`. You can also assign the same initial value to a group of parameters. For example, `initial({a b c} 1)` will initialize parameters `{a}`, `{b}`, and `{c}` to 1 and `initial({lc:x1 x2 _cons} 0)` will initialize `{lc:x1}`, `{lc:x2}`, and `{lc:_cons}` to 0. You can assign the same initial value to all parameters with the same group name. For example, we can shorten the previous specification to `initial({lc:} 0)`.

Depending on the situation, it may also be beneficial to specify initial values for the NLS algorithm used by `menl` to obtain starting values for the EM algorithm. These initial values can be specified in the parameter definition such as `{a=0.5}`, in which case the NLS algorithm used during the initialization will use 0.5 as the starting value for parameter `a` instead of the default 0. Such initialization is particularly useful for parameters used in the denominators for which zero values may lead to an undefined value of the mean function.

See *Examples of specifying initial values* and *Obtaining initial values* for examples.

Two-level models

The sole purpose of this section and its examples is to highlight the syntax of `menl` and make you familiar with how to specify substitutable expressions in `menl` and with its output. Also see an introductory example in *Nonlinear models* in [ME] **me**.

We will use the data from the Longitudinal Study of Unicorn Health in Zootopia, which contain the brain weight (`weight`) of 30 newborn male unicorns and 30 newborn female unicorns. Measurements were collected at 13 occasions every 2 months over the first 2 years after birth (`time`). Based on previous studies, a model for unicorn brain shrinkage is believed to be

$$\texttt{weight}_{ij} = \beta_1 + (\beta_2 - \beta_1)\exp(-\beta_3\texttt{time}_{ij}) + \epsilon_{ij} \quad i = 1, 2, \ldots, 13;\ j = 1, 2, \ldots, 60$$

Parameter β_1 represents the average brain weight of unicorns as $\texttt{time}_{ij}$ increases to infinity. Parameter β_2 is the average brain weight at birth (at $\texttt{time}_{ij} = 0$), and β_3 is a scale parameter that determines the rate at which the average brain weight of unicorns approaches the asymptotic weight β_1 (decay rate). This model can be fit with the `nl` command; see [R] **nl**.

We will start with a simple two-level model in which we allow the asymptote parameter β_1 to vary between unicorns by replacing β_1 in the above equation with $\beta_1 + u_{0j}$,

$$\texttt{weight}_{ij} = \beta_1 + u_{0j} + (\beta_2 - \beta_1 - u_{0j})\exp(-\beta_3\texttt{time}_{ij}) + \epsilon_{ij} \tag{4}$$

where β_1, β_2, and β_3 are fixed-effects parameters to be estimated and u_{0j} is a random intercept at the unicorn, `id`, level that follows a normal distribution with mean 0 and variance σ_u^2.

Equivalently, the model defined by (4) can be written as a two-stage model,

$$\texttt{weight}_{ij} = \phi_{1j} + (\phi_{2j} - \phi_{1j})\exp(-\phi_{3j}\texttt{time}_{ij}) + \epsilon_{ij} \tag{5}$$

with the following stage 2 specification:

$$\begin{aligned}\phi_{1j} &= \beta_1 + u_{0j}\\ \phi_{2j} &= \beta_2\\ \phi_{3j} &= \beta_3\end{aligned} \tag{6}$$

Parameters ϕ_{1j}, ϕ_{2j}, and ϕ_{3j} now describe the behavior of the jth unicorn. For example, ϕ_{1j} represents the brain weight of the jth unicorn as $\texttt{time}_{ij}$ increases to infinity.

▷ Example 1: Simple two-level model

Let's use `menl` to first fit a single-equation model defined by (4), described above.

```
. use http://www.stata-press.com/data/r15/unicorn
(Brain shrinkage of unicorns in the land of Zootopia)
. menl weight = {b1}+{U0[id]}+({b2}-{b1}-{U0[id]})*exp(-{b3}*time)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -56.9757597
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
Linearization log likelihood =  -56.97576
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b1	4.707954	.1414511	33.28	0.000	4.430715	4.985193
/b2	8.089432	.0260845	310.12	0.000	8.038307	8.140556
/b3	4.13201	.0697547	59.24	0.000	3.995293	4.268726

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Identity				
var(U0)	1.189809	.2180063	.8308322	1.703888
var(Residual)	.0439199	.0023148	.0396095	.0486995

Notes:

1. The response variable `weight` is specified on the left-hand side of the equality sign, and parameters to be estimated are enclosed in curly braces `{b1}`, `{b2}`, and `{b3}` on the right-hand side.
2. By typing `{U0[id]}`, we specified a random intercept at the level identified by the group variable `id`, that is, the unicorn level (level two).

3. The estimation log consists of three parts:
   a. A set of EM iterations used to refine starting values. By default, the iterations themselves are not displayed, but you can display them by using the `emlog` option. NLME models may often have multiple solutions and converge to a local maximum. It is thus important to try different initial values to investigate the existence of multiple solutions and the convergence to a global maximum; see *Obtaining initial values*.
   b. A set of iterations displaying the value of the linearization log likelihood from the Lindstrom–Bates algorithm or alternating algorithm. The term "linearization" reflects the fact that the reported log likelihood corresponds to the linear mixed-effects model obtained after linearization of the specified nonlinear mean function with respect to fixed and random effects. See *Inference based on linearization* and *Stopping rules* for details about the algorithm.
   c. The message "Computing standard errors". This is just to inform you that `menl` has finished its iterative maximization and is now reparameterizing the variance components (see *Methods and formulas*) to their natural metric and computing their standard errors. If you are interested only in the fixed-effects estimates, you can use the `nostderr` option to bypass this step.
4. The output title, "Mixed-effects ML nonlinear regression", informs us that our model was fit using ML, the default. For REML estimates, use the `reml` option.
5. The header information is similar to that of the `mixed` command, but unlike `mixed`, `menl` in general does not report a model χ^2 statistic in the header because a test of the joint significance of all fixed-effects parameters (except the constant term) may not be relevant in a nonlinear model.
6. The first estimation table reports the fixed effects. We estimate $\beta_1 = 4.71$, $\beta_2 = 8.09$, and $\beta_3 = 4.13$. Although z tests against zeros are reported automatically for all fixed-effects parameters, as part of standard estimation output, they may not always be of interest or even appropriate for parameters of nonlinear models. You can use the `test` command ([R] **test**) to test hypotheses of interest or reparameterize your model so that the tests of parameters against zeros are meaningful.
7. The second estimation table shows the estimated variance components. The first section of the table is labeled `id: Identity`. In general, this means that our model includes random effects at the `id` (unicorn) level and that their variance–covariance matrix, $\mathbf{\Sigma}$, is the identity matrix (all random effects have the same variance). In our example, because we have only one random effect, u_{0j}, the random-effect covariance structure is irrelevant, and the variance of the random intercept, σ_u^2, labeled as `var(U0)` in the output, is estimated as 1.19 with standard error 0.22.
8. The row labeled `var(Residual)` displays the estimated overall error variance or variance of the error term; that is, $\widehat{\mathrm{Var}}(\epsilon_{ij}) = \widehat{\sigma}_\epsilon^2 = 0.044$.

◁

▷ Example 2: Two-level model as a two-stage model, using the define() option

The model from example 1 can also be specified as a two-stage model, as defined by (5) and (6), by using the `define()` option. The `define()` option is particularly useful when you have a complicated nonlinear expression, and you would like to break it down into smaller pieces. Parameters of interest that are functions of other parameters can be defined using the `define()` option. This will make it easier to predict them for each subject after estimation; see [ME] **menl postestimation**.

Below we specify the asymptote parameter, ϕ_{1j}, by using `define()`. The colon (`:`) in `{phi1:}` instructs `menl` that `phi1` will be specified as a substitutable expression within the `define()` option. Parameters `{phi2}` and `{phi3}` are simple free parameters and thus do not need to be specified in `define()`.

```
. menl weight = {phi1:}+({phi2}-{phi1:})*exp(-{phi3}*time),
> define(phi1: {b1}+{U0[id]})
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -56.9757597
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
Linearization log likelihood =  -56.97576
        phi1:  {b1}+{U0[id]}
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b1	4.707954	.1414511	33.28	0.000	4.430715	4.985193
/phi2	8.089432	.0260845	310.12	0.000	8.038307	8.140556
/phi3	4.13201	.0697547	59.24	0.000	3.995293	4.268726

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Identity				
var(U0)	1.189809	.2180058	.8308328	1.703887
var(Residual)	.0439199	.0023148	.0396095	.0486995

The results are identical to those obtained in example 1, but the estimation table now has a legend that lists the expression `phi1` defined in the model. We can suppress this legend by specifying the `nolegend` option.

We could have defined `phi1` directly in the main expression instead of the `define()` option,

```
. menl weight = {phi1:{b1}+{U0[id]}}+({phi2}-{phi1:})*exp(-{phi3}*time)
(output omitted)
```

but by using the `define()` option, we simplified the main expression.

◁

▷ Example 3: Two-level model containing covariates

Reducing brain weight loss has been an active research area in Zootopia for the past two decades, and scientists believe that consuming rainbow cupcakes right after birth may help slow down brain shrinkage. Recall that the scale parameter ϕ_{3j} determines the rate at which the brain weight of the jth unicorn decreases to its asymptotic value ϕ_{1j}. Hence, covariate `cupcake`, which represents the number of rainbow cupcakes consumed right after birth, is added to the equation of ϕ_{3j}. Also, we would like to investigate whether the asymptote, ϕ_{1j}, is gender specific, so we include the factor variable `female` in the equation for ϕ_{1j}. $\texttt{female}_j$ is 1 if the jth unicorn is a female and 0 otherwise.

The stage 2 specification of the model defined by (5) becomes

$$\begin{aligned}\phi_{1j} &= \beta_{10} + \beta_{11}\texttt{female}_j + u_{0j}\\ \phi_{2j} &= \beta_2\\ \phi_{3j} &= \beta_{30} + \beta_{31}\texttt{cupcake}_j\end{aligned} \tag{7}$$

The `define()` option can be repeated, so we specify a separate `define()` option for ϕ_{1j}, ϕ_{2j}, and ϕ_{3j}. We could have left ϕ_{2j} as a free parameter `{phi2}` in our specification, but we wanted to closely match the stage 2 specification (7).

```
. menl weight = {phi1:}+({phi2:}-{phi1:})*exp(-{phi3:}*time),
> define(phi1: {b10}+{b11}*1.female+{U0[id]})
> define(phi2: {b2})
> define(phi3: {b30}+{b31}*cupcake)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -29.0149875

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60

                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13

Linearization log likelihood = -29.014988

        phi1:  {b10}+{b11}*1.female+{U0[id]}
        phi2:  {b2}
        phi3:  {b30}+{b31}*cupcake
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b10	4.072752	.1627414	25.03	0.000	3.753785	4.39172
/b11	1.264407	.2299723	5.50	0.000	.8136694	1.715144
/b2	8.088102	.0255465	316.60	0.000	8.038032	8.138172
/b30	4.706926	.1325714	35.50	0.000	4.44709	4.966761
/b31	-.2007309	.0356814	-5.63	0.000	-.2706651	-.1307966

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Identity				
var(U0)	.7840578	.1438924	.5471839	1.123473
var(Residual)	.0420763	.0022176	.0379468	.0466551

In the table legend, `/b10` and `/b11` correspond, respectively, to the constant term and coefficient of `1.female` in the equation for ϕ_{1j}. `/b2` is ϕ_{2j}, and `/b30` and `/b31` correspond, respectively, to the constant term and coefficient for `cupcake` in the equation for ϕ_{3j}.

Based on our results, consuming rainbow cupcakes after birth indeed slows down brain shrinkage: `/b31` is roughly -0.2 with a 95% CI of $[-0.271, -0.131]$.

◁

▷ Example 4: Specifying linear combinations

A more convenient way to specify the model in example 3 is to use linear-combination specifications; see *Random-effects substitutable expressions*.

For example, `define(phi1: {b10}+{b11}*1.female+{U0[id]})` can be replaced with `define(phi1: i.female U0[id])`. `menl` knows that we are defining ϕ_{1j} as a linear combination of `i.female` and `U0[id]` and thus will create a constant term and a coefficient for each level of factor

variable female and will use a coefficient of 1 for any random effect. Because female has only two levels, menl will create two coefficients for 0b.female and 1.female, respectively, but will constrain the coefficient of the base level, level 0, to be 0.

We now fit our model by using linear-combination specifications within the define() options. We explain the use of the second and third define() specifications following estimation.

```
. menl weight = {phi1:}+({phi2:}-{phi1:})*exp(-{phi3:}*time),
> define(phi1: i.female U0[id])
> define(phi2: _cons, xb)
> define(phi3: cupcake, xb)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -29.0149875

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60

                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13

                                                Wald chi2(2)      =      61.78
Linearization log likelihood = -29.014987       Prob > chi2       =     0.0000

        phi1:  i.female U0[id]
        phi3:  cupcake, xb
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
female						
female	1.264407	.2299723	5.50	0.000	.8136694	1.715144
_cons	4.072752	.1627414	25.03	0.000	3.753785	4.39172
phi2						
_cons	8.088102	.0255465	316.60	0.000	8.038032	8.138172
phi3						
cupcake	-.2007309	.0356814	-5.63	0.000	-.2706651	-.1307966
_cons	4.706926	.1325714	35.50	0.000	4.44709	4.966761

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Identity				
var(U0)	.7840578	.1438928	.5471833	1.123475
var(Residual)	.0420763	.0022176	.0379468	.0466551

By using linear-combination specifications within the define() options, we improved the readability of the coefficient table. For example, it is now clear that _cons in the equation labeled phi3 corresponds to the constant term in the equation for ϕ_{3j}. This term was labeled /b30 previously.

Notes:

1. The define() option interprets its specification as a random-effects substitutable expression, so you do not need to specify the curly braces ({}) around the specification.
2. All rules for random-effects substitutable expressions apply to the specifications within define().

3. Following one of the rules for random-effects substitutable expressions, we used the `xb` option within `define()`s for `phi2` and `phi3`, because their linear combinations contained only one term: `_cons` for `phi2` and `cupcake` for `phi3`.
4. Specification `define(phi2: _cons, xb)` is the same as `define(phi2:, xb)`. In other words, `_cons` is implied with any linear combination, unless the `noconstant` option is specified. We chose to include `_cons` directly for clarity.
5. We could have used a free parameter `{phi2}` instead of the linear combination `{phi2: _cons, xb}`, but we wanted to preserve the order in which `phi1`, `phi2`, and `phi3` appear in the estimation table. See example 5, where we specify ϕ_{2j} as a free parameter `{phi2}`.
6. In the presence of linear combinations, `menl` reports a joint test of significance of all coefficients (except the constant term) across all linear combinations.
7. Linear combinations containing only a constant such as `{phi2:}` are not listed in the table expression legend for brevity.

◁

▷ Example 5: Including random coefficients

In previous examples, we only illustrated how to specify random intercepts such as `{U0[id]}`, and it is bad karma to end a unicorn story without showing how to specify random coefficients or random slopes.

Continuing with our model as defined by (5) and (7), let's suppose that the equation for the brain-weight scale parameter, ϕ_{3j}, is as follows:

$$\phi_{3j} = \beta_{30} + (\beta_{31} + u_{1j})\texttt{cupcake}_j$$

We incorporated a unicorn-specific random slope for variable `cupcake`. The random slope, u_{1j}, for a continuous variable `cupcake` can be specified in `menl` as `c.cupcake#U1[id]`, and by default, `menl` assumes that it is independent of the random intercept, u_{0j}. (See example 9 for specifying other random-effects covariance structures.)

```
. menl weight = {phi1:}+({phi2}-{phi1:})*exp(-{phi3:}*time),
> define(phi1: i.female U0[id])
> define(phi3: cupcake c.cupcake#U1[id])

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = 165.417513
Iteration 2:    linearization log likelihood = 165.420082
Iteration 3:    linearization log likelihood = 165.420112
Iteration 4:    linearization log likelihood = 165.420101

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
                                                Wald chi2(2)      =      46.70
Linearization log likelihood =   165.4201       Prob > chi2       =     0.0000

        phi1:  i.female U0[id]
        phi3:  cupcake c.cupcake#U1[id]
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
female						
female	1.320623	.2215707	5.96	0.000	.8863522	1.754893
_cons	4.006823	.1568268	25.55	0.000	3.699448	4.314198
phi3						
cupcake	-.219661	.0659984	-3.33	0.001	-.3490155	-.0903066
_cons	4.771466	.1128421	42.28	0.000	4.5503	4.992633
/phi2	8.087655	.0179406	450.80	0.000	8.052492	8.122818

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
id: Independent				
var(U0)	.727464	.1337149	.5074024	1.042967
var(U1)	.1258914	.0309569	.0777471	.2038487
var(Residual)	.0208202	.0011403	.018701	.0231795

In addition to the overall error variance and the random-intercept variance, we now have a random-slope variance, which is labeled `var(U1)` in the output. In this example, we also specified parameter ϕ_{2j} as a free parameter `{phi2}` instead of a linear combination as in example 4. As we mentioned in *Summary*, free parameters are displayed after linear combinations, so `phi2` is listed last in the estimation table.

Previous studies of unicorns considered a model that also incorporated gender-specific variation between unicorns in asymptotic weight ϕ_{1j},

$$\phi_{1j} = \beta_{10} + u_{0j} + (\beta_{11} + u_{2j})\texttt{female}_j$$

but found no statistical evidence of such variation.

If we wanted to verify this for our data, we could have fit the following model:

```
. menl weight = {phi1:}+({phi2}-{phi1:})*exp(-{phi3:}*time), ///
    define(phi1: i.female U0[id] 1.female#U2[id])           ///
    define(phi3: cupcake c.cupcake#U1[id])
```

Compared with our previous specification, we included a new term in the `define()` option for `phi1`—a random slope for level 1 of the factor variable `female`, `1.female#U2[id]`. To include random slopes for a factor variable, we must specify random effects for each level, except the base level, of the factor variable. The specification `i.`*fvvarname* for referring to all levels of a factor variable is not allowed in the context of random effects, because a different set of random effects must be defined for each level. For example, if we specified `i.female#U2[id]` in our example, we would have received an error.

◁

To summarize:

1. Use `{`*name*`}` to define free parameters such as `{b1}`.
2. Use, for example, `{U0[id]}` to define random intercepts at the `id` level, `{c.`*varname*`#U1[id]}` to define random slopes for continuous variable *varname* at the `id` level, and `{#.`*fvvarname*`#U1[id]}` for each level *#*, except the base level, of variable *fvvarname* to include random slopes for factor variable *fvvarname*. The specification `{i.`*fvvarname*`#U1[id]}` is not allowed.
3. Use linear-combination specifications whenever possible. Do not use `{}` around random effects when they are specified within a linear combination.
4. Use multiple `define()` options to specify parameters of interest that are functions of other parameters, and use linear-combination specifications within `define()` whenever possible.
5. Use the `xb` option within a linear combination or within `define()` whenever you specify one variable such as `define(phi1: cupcake, xb)`, one random effect such as `{phi2: U0[id], xb}`, or a constant-only linear combination such as `{phi2: _cons, xb}` or `{phi2: , xb}`. When you specify the `xb` option, the above specifications are interpreted by `menl`, respectively, as a linear combination `{phi1:_cons}+{phi1:cupcake}*cupcake`, a linear combination `{phi:_cons}+{U0[id]}`, and a constant term `{phi2:_cons}`.
6. Unicorns do exist in our world, they are just gray, fat, and called rhinos.

Testing variance components

Consider data on the intensity of 23 large earthquakes in western North America between 1940 and 1980 analyzed originally by Joyner and Boore (1981) and then also by Davidian and Giltinan (1995, sec. 11.4). The objective of the study was to model the maximum horizontal acceleration (in g units), `accel`, taken at the ith measuring station for the jth earthquake, as a function of the magnitude of the quake on the Richter scale, `richter`, and the distance (in km) of the measuring station from the quake epicenter, `distance`. We are also interested in the possible effect of the soil type `soil`, soil versus rock, at a given measuring station on acceleration. The results of this study are useful to understand the nature of the damage caused by a particular earthquake and to determine the location for sensitive installations such as nuclear facilities.

Let's consider one of the models studied by Davidian and Giltinan (1995) for these data,

$$\log_{10}(\texttt{accel}_{ij}) = \phi_{1j} - \log_{10}\sqrt{\texttt{distance}_{ij}^2 + \exp(\phi_{2j})} - \phi_{3ij}\sqrt{\texttt{distance}_{ij}^2 + \exp(\phi_{2j})} + \epsilon_{ij} \tag{8}$$

where

$$\begin{aligned}\phi_{1j} &= \beta_0 + \beta_1 \texttt{richter}_j + u_{1j} \\ \phi_{2j} &= \beta_2 \\ \phi_{3i} &= \beta_3 + u_{3j}\end{aligned} \tag{9}$$

and

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{3j} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}), \text{ diagonal } \mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{u_1} & 0 \\ 0 & \sigma^2_{u_3} \end{bmatrix}, \text{ and } \epsilon_{ij} \sim N(0, \sigma^2_\epsilon) \tag{10}$$

▷ Example 6: Fitting an NLME model for the earthquake data

Let's fit the model defined by (8), (9), and (10) by using `menl`.

```
. use http://www.stata-press.com/data/r15/earthquake
(Earthquake intensity (Joyner and Boore, 1981))
. menl laccel = {phi1:}-log10(sqrt(c.distance#c.distance+exp({phi2})))
> -{phi3:}*sqrt(c.distance#c.distance+exp({phi2})),
> define(phi1: richter U1[quake]) define(phi3: U3[quake], xb)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = 2.4115811
Iteration 2:    linearization log likelihood = 2.40751409
Iteration 3:    linearization log likelihood = 2.40734699
Iteration 4:    linearization log likelihood = 2.40734238
Iteration 5:    linearization log likelihood = 2.40734121
Iteration 6:    linearization log likelihood = 2.40734113
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        182
Group variable: quake                           Number of groups  =         23
                                                Obs per group:
                                                              min =          1
                                                              avg =        7.9
                                                              max =         38
                                                Wald chi2(1)      =      26.26
Linearization log likelihood =  2.4073411       Prob > chi2       =     0.0000
        phi1:  richter U1[quake]
        phi3:  U3[quake], xb
```

laccel	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
richter	.2310021	.0450804	5.12	0.000	.1426461	.319358
_cons	-.8836537	.2826255	-3.13	0.002	-1.437589	-.329718
phi3						
_cons	.004575	.0014192	3.22	0.001	.0017935	.0073566
/phi2	4.063075	.4023386	10.10	0.000	3.274506	4.851644

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
quake: Independent				
var(U1)	.0056676	.0073404	.0004477	.071752
var(U3)	.000013	8.42e-06	3.66e-06	.0000463
var(Residual)	.0461647	.0054421	.0366409	.0581639

We also store our estimates for later use:

```
. estimates store E1
```

By default, `menl` assumes that the random effects u_{1j} and u_{3j} are independent, so there is no need to specify the `covariance()` option in this case. In other words, omitting the `covariance()` option is equivalent to specifying `covariance(U1 U3, independent)`.

◁

▷ Example 7: Likelihood-ratio test for variance components

Davidian and Giltinan (1995) did not include any random effects in the model for the ϕ_{2j} parameters. Let's check whether the random effects are needed in the equations for ϕ_{1j} and ϕ_{3j} parameters in (9).

One simple way to assess whether a random effect associated with a certain ϕ_j can be omitted, is to examine its coefficient of variation (CV), the ratio of the standard deviation to the mean. Let's compute the CV for ϕ_{3j}. For convenience, let's redisplay the results from example 6 as standard deviations for variance components.

```
. menl, stddeviations
Mixed-effects ML nonlinear regression          Number of obs     =        182
Group variable: quake                          Number of groups  =         23
                                               Obs per group:
                                                             min =          1
                                                             avg =        7.9
                                                             max =         38
                                               Wald chi2(1)      =      26.26
Linearization log likelihood =  2.4073411      Prob > chi2       =     0.0000
        phi1:  richter U1[quake]
        phi3:  U3[quake], xb
```

laccel	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
richter	.2310021	.0450804	5.12	0.000	.1426461	.319358
_cons	-.8836537	.2826255	-3.13	0.002	-1.437589	-.329718
phi3						
_cons	.004575	.0014192	3.22	0.001	.0017935	.0073566
/phi2	4.063075	.4023386	10.10	0.000	3.274506	4.851644

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
quake: Independent				
sd(U1)	.0752832	.0487517	.0211582	.2678656
sd(U3)	.0036085	.0011673	.0019142	.0068026
sd(Residual)	.2148596	.0126644	.1914181	.241172

The `stddeviations` option specifies that `menl` display random-effects and error standard deviations instead of variances. It will also display correlations instead of covariances whenever they are in the model. Because random-effects variances for these data are very small, we will use this option in all subsequent examples to display results in the standard deviation metric.

The interquake random variation in the ϕ_{3j} values about their mean is CV = `sd(U3)/{phi3:_cons}` $= 0.0036/0.0046 \approx 78\%$, and it appears reasonable to keep it in the model. You can perform a formal likelihood-ratio (LR) test of H_0: $\sigma^2_{u_3} = 0$ to verify this, as we show below for the test of H_0: $\sigma^2_{u_1} = 0$.

Let's check whether we need random intercept u_{1j} to model ϕ_{1j}. Computing CV in this case to get an initial assessment is not simple because the mean of ϕ_{1j} depends on the jth quake through variable `richter`. Given the same main equation (8), we will use the LR test to compare the restricted model, with u_{1j} excluded, which is defined by (11) and (12) below, with the full model defined by (9) and (10).

The stage 2 specification of the restricted model is

$$\begin{aligned} \phi_{1j} &= \beta_0 + \beta_1 \texttt{richter}_j \\ \phi_{2j} &= \beta_2 \\ \phi_{3ij} &= \beta_3 + u_{3j} \end{aligned} \tag{11}$$

where

$$u_{3j} \sim N(0, \sigma^2_{u_3}) \quad \text{and} \quad \epsilon_{ij} \sim N(0, \sigma^2_{\epsilon}) \tag{12}$$

We now fit the restricted model:

```
. menl laccel = {phi1:}-log10(sqrt(c.distance#c.distance+exp({phi2})))
> -{phi3:}*sqrt(c.distance#c.distance+exp({phi2})),
> define(phi1: richter, xb) define(phi3: U3[quake], xb)
> stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = 2.12628622
Iteration 2:    linearization log likelihood = 2.126043
Iteration 3:    linearization log likelihood = 2.12603276
Iteration 4:    linearization log likelihood = 2.12603003
Iteration 5:    linearization log likelihood = 2.1260297

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        182
Group variable: quake                           Number of groups  =         23
                                                Obs per group:
                                                              min =          1
                                                              avg =        7.9
                                                              max =         38
                                                Wald chi2(1)      =      32.22
Linearization log likelihood =  2.1260297       Prob > chi2       =     0.0000

        phi1:  richter, xb
        phi3:  U3[quake], xb
```

laccel	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
richter	.2208878	.0389144	5.68	0.000	.1446169	.2971586
_cons	-.7863293	.2503442	-3.14	0.002	-1.276995	-.2956637
phi3						
_cons	.0054348	.0015661	3.47	0.001	.0023653	.0085044
/phi2	4.228431	.3702251	11.42	0.000	3.502803	4.954059

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
quake: Identity				
sd(U3)	.0042144	.0011309	.0024907	.0071309
sd(Residual)	.2170084	.0122821	.1942231	.2424668

```
. estimates store E2
```

Next, we use `lrtest` to perform an LR test of the hypothesis:

$$H_0\colon \sigma^2_{u_1} = 0 \qquad \text{versus} \qquad H_1\colon \sigma^2_{u_1} \neq 0$$

```
. lrtest E1 E2, stats
Likelihood-ratio test                                 LR chi2(1)  =      0.56
(Assumption: E2 nested in E1)                         Prob > chi2 =    0.4532
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space.  If this is not true, then the
      reported test is conservative.
Akaike's information criterion and Bayesian information criterion
```

Model	Obs	ll(null)	ll(model)	df	AIC	BIC
E2	182	.	2.12603	6	7.747941	26.97198
E1	182	.	2.407341	7	9.185318	31.61336

```
Note: N=Obs used in calculating BIC; see [R] BIC note.
```

Because testing of H_0: $\sigma^2_{u_1} = 0$ is on the boundary of the parameter space, `lrtest` reports a note that the provided LR test is conservative; that is, the actual p-value is smaller than the one reported. For a test of H_0: $\sigma^2_{u_1} = 0$ in a two-level model, the true asymptotic distribution is not $\chi^2(1)$ but a mixture of $\chi^2(0)$ and $\chi^2(1)$ with equal weights, $0.5\chi^2(0) + 0.5\chi^2(1)$; thus the p-value is actually $0.4532/2 = 0.2266$ (see Rabe-Hesketh and Skrondal 2012, sec 8.8). We do not have sufficient evidence to reject the null hypothesis, so we can omit random effect u_{1j} from the full model. AIC and BIC also favor a simpler, reduced model.

◁

▷ Example 8: Including within-subject covariates

One of the questions of interest in the earthquake study was the potential effect of the soil type on acceleration. Variable `soil` is a within-subject covariate because the values $\texttt{soil}_{ij}$ may vary within a subject (earthquake). We include variable `soil` in the equation for ϕ_{3ij} in (11),

$$\begin{aligned}
\phi_{1j} &= \beta_0 + \beta_1 \texttt{richter}_j \\
\phi_{2j} &= \beta_2 \\
\phi_{3ij} &= \beta_3 + \beta_4 \texttt{soil}_{ij} + u_{3j}
\end{aligned}$$

and fit the corresponding model:

```
. menl laccel = {phi1:}-log10(sqrt(c.distance#c.distance+exp({phi2})))
> -{phi3:}*sqrt(c.distance#c.distance+exp({phi2})),
> define(phi1: richter, xb) define(phi3: i.soil U3[quake]) stddeviations
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = 3.56347788
Iteration 2:    linearization log likelihood = 3.56324717
Iteration 3:    linearization log likelihood = 3.56323391
Iteration 4:    linearization log likelihood = 3.56323036
Iteration 5:    linearization log likelihood = 3.56322978
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        182
Group variable: quake                           Number of groups  =         23
                                                Obs per group:
                                                              min =          1
                                                              avg =        7.9
                                                              max =         38
                                                Wald chi2(2)      =      34.20
Linearization log likelihood =  3.5632298       Prob > chi2       =     0.0000
        phi1:  richter, xb
        phi3:  i.soil U3[quake]
```

laccel	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
richter	.2275944	.0395549	5.75	0.000	.1500683	.3051206
_cons	-.8079826	.2548833	-3.17	0.002	-1.307545	-.3084205
phi3						
soil						
soil	-.0011041	.0006441	-1.71	0.087	-.0023665	.0001583
_cons	.0067347	.0017416	3.87	0.000	.0033213	.0101481
/phi2	4.3212	.3653809	11.83	0.000	3.605067	5.037334

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
quake: Identity				
sd(U3)	.0043088	.0011285	.0025788	.0071992
sd(Residual)	.2147101	.0121424	.1921829	.2398779

The estimated coefficient for the soil type is -0.0011 with a 95% CI of $[-0.0024, 0.0002]$. The knowledge of the soil type at a particular site does not appear to add explanatory power to our model. ◁

Random-effects covariance structures

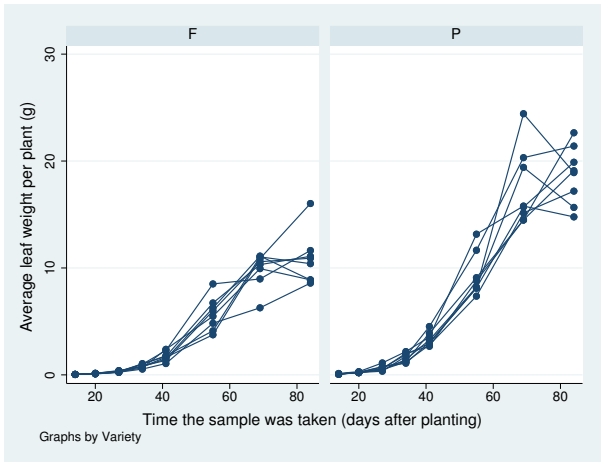
`menl` supports various covariance structures to model the random-effects covariance matrix. They are specified using the `covariance()` option. The `covariance()` option may be repeated. This is necessary to accommodate multilevel NLME models, where you may need to specify different covariance matrices for the random effects at different levels. Repeating this option may also be useful if you want to specify a block-diagonal covariance structure. See example 19 for details.

▷ Example 9: Two-level model with correlated random effects

Davidian and Giltinan (1995, sec. 1.1.3 and 11.2) discuss a study of soybean plants that started in 1988 and spanned over three growing seasons, `year`. The central objective of the study was to compare the growth patterns of two genotypes of soybean plants, `variety`: a commercial variety of soybean, denoted by F, and an experimental variety, denoted by P. In each season, eight plots were planted using F variety and eight using P variety. To assess growth, researchers sampled each plot 8 to 10 times ($8 \leq n_j \leq 10$) at approximately weekly intervals, `time`. At each sampling time, six plants were taken from each plot at random. Leaves from the plants were weighed, and the resulting total weight was divided by six to yield a measure of the average leaf weight per plant (in g) for the plot for that week, `weight`. Plots are identified by the `plot` variable.

Let's plot the data first.

```
. use http://www.stata-press.com/data/r15/soybean
(Growth of soybean plants (Davidian and Giltinan, 1995))
. twoway connected weight time if year==2, connect(L) by(variety)
```



The graph shows the average leaf weights per plant over time for the eight plots with plants of each genotype in the 1989 growing season. Longitudinal growth measures for each plot are connected with solid lines. Apart from some intraplot variation, the growth profile of each plot follows roughly an S shape, according to which growth begins slowly, then shows a linear trend during the middle of the growing season, and then "levels off" at the end. Such pattern is typical for many growth studies.

The main goal of the study was to compare growth patterns over the growing season for the two soybean genotypes. Because the three growing seasons differed markedly in terms of precipitation—1988 was unusually dry, 1989 was wet, and 1990 was normal—contrasting these growth patterns across years was also of interest. The results of this study are useful, for example, for harvesting purposes.

A popular model for individual profiles that resemble an S shape is the logistic growth model:

$$\texttt{weight}_{ij} = \frac{\phi_{1j}}{1 + \exp\left\{-\left(\texttt{time}_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij} \tag{13}$$

ϕ_{1j} is the asymptotic average leaf weight per soybean plant in plot j as $\texttt{time}_{ij} \to \infty$. ϕ_{2j} is the time at which half of ϕ_{1j} is reached; that is, if $\texttt{time}_{ij} = \phi_{2j}$, then $E(\texttt{weight}_{ij}) = 0.5\phi_{1j}$. ϕ_{1j}

and ϕ_{2j} will henceforth be referred to as "the limiting growth" and "half-life", respectively. ϕ_{3j} is a scale parameter, and it represents the number of days it takes for average leaf weight to grow from 50% (half-life) to about 73% of its limiting growth. That is, if we set $\texttt{time}_{ij} = t_{0.73} = \phi_{2j} + \phi_{3j}$, the right-hand side of (13), ignoring the error term, reduces to $\phi_{1j}/\{1 + \exp(-1)\} = 0.73\phi_{1j}$, and then $\phi_{3j} = t_{0.73} - \phi_{2j}$.

We will start with a simple stage 2 specification that does not contain any covariates. Also, because the number of soybean plots, 48, is large compared with the number of random effects, 3, we consider a general positive-definite, unstructured, random-effects covariance matrix:

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \tag{14}$$

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}), \; \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} \end{bmatrix}, \; \epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$$

To specify this covariance structure in `menl`, we specify `unstructured` in the `covariance()` option. The `covariance()` option also requires that we list the names of random effects to be correlated.

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb) define(phi3: U3[plot], xb)
> covariance(U1 U2 U3, unstructured)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -739.901421
Iteration 2:    linearization log likelihood = -739.84929
  (iteration log omitted)
Iteration 39:   linearization log likelihood = -739.834521
Iteration 40:   linearization log likelihood = -739.834448

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10

Linearization log likelihood = -739.83445

        phi1:  U1[plot], xb
        phi2:  U2[plot], xb
        phi3:  U3[plot], xb
```

weight		Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1							
	_cons	19.25314	.8031811	23.97	0.000	17.67893	20.82734
phi2							
	_cons	55.01999	.7272491	75.65	0.000	53.59461	56.44537
phi3							
	_cons	8.403468	.3152551	26.66	0.000	7.78558	9.021357

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
plot: Unstructured				
var(U1)	27.05081	6.776516	16.55561	44.19929
var(U2)	17.61605	5.317899	9.748766	31.83227
var(U3)	1.972036	.9849825	.7409021	5.248904
cov(U1,U2)	15.73304	5.413365	5.123042	26.34304
cov(U1,U3)	5.193819	2.165586	.9493488	9.438289
cov(U2,U3)	5.649306	2.049458	1.632442	9.66617
var(Residual)	1.262237	.1111686	1.062119	1.50006

The expected limiting growth or expected maximum average weight, $\beta_1 = E(\phi_{1j})$, of soybean leaves is estimated to be around 19.25 grams. The expected half-life or the time at which the leaves reach half of their expected maximum average weight, $\beta_2 = E(\phi_{2j})$, is estimated to be around 55 days after planting. The expected time needed for the average leaf weight per plant to grow from 50% to 73% of the limiting growth, $\beta_3 = E(\phi_{3j})$, is about 8.4 days.

The estimates of the six random-effects variance–covariance parameters σ_{11}, σ_{22}, σ_{33}, σ_{12}, σ_{13}, and σ_{23} are displayed in the upper part of the random-effects parameters table. There is a plot-to-plot variation in the estimates of all three parameters of interest: β_1, β_2, and β_3. Also, the plot-specific effects associated with the parameters of interest are positively correlated. For example, based on the estimate of 5.19 of `cov(U1,U3)`, plants with larger maximum weights tend to grow faster.

We store our estimates for later use:

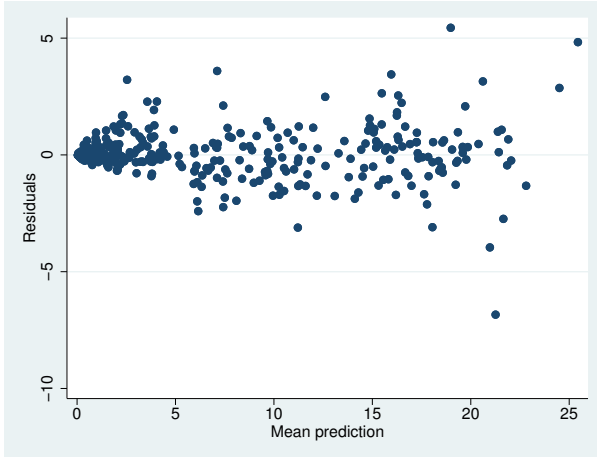
```
. estimates store S1
```

◁

▷ Example 10: Residuals-vs-fitted plot to check for heteroskedasticity

A popular tool for investigating within-cluster heteroskedasticity is the plot of residuals against the predicted values and other candidate variance covariates. For growth models, variance is often a function of the mean (predicted values). Below we construct the plot of residuals versus predicted values to evaluate the assumption of homoskedastic errors in example 9.

```
. predict fitweight, yhat
. predict res, residuals
. scatter res fitweight
```



The plot reveals increasing variability with the predicted average leaf weights, which indicates that our within-cluster variance model is misspecified. In *Heteroskedastic within-group errors*, we will show how to account for within-cluster heteroskedasticity by using the `resvariance()` option.

◁

Heteroskedastic within-group errors

Until now, we assumed that the within-group errors—the ϵ's in the considered models—are i.i.d. Gaussian with common variance σ^2_ϵ, labeled as `var(Residual)` by `menl` in the output.

To relax the assumptions of homoskedasticity and the independence of errors, `menl` provides two alternatives. You can model the within-group error variance–covariance matrix, $\sigma^2\mathbf{\Lambda}_j$, directly by using the `rescovariance()` option. If you used the `mixed` command and its `residuals()` option before, you should be familiar with this approach. Alternatively, you can model the error variance–covariance matrix indirectly by modeling the heteroskedasticity structure with the `resvariance()` option and the correlation structure with the `rescorrelation()` option; see *Variance-components parameters*. The latter approach offers more flexibility, particularly in modeling the heteroskedasticity structure. For example, many NLME models exhibit within-subject heterogeneity that is a power function of the mean. The `rescovariance()` option cannot model this, but `resvariance(power _yhat)` can.

If your error structure is simple and is similar to those encountered in `mixed`, you can use the `rescovariance()` option. Otherwise, use `resvariance()`, `rescorrelation()`, or both to model more flexible within-group error covariance structures.

▷ Example 11: Heteroskedastic power structure

Continuing with example 9, for these types of growth data, we find it is common for the intraplot variance to increase systematically with the average leaf weight, as we saw in example 10 from the residuals-versus-fitted plot. Davidian and Giltinan (1995) proposed a variance structure that models the within-group error variance as a power function of the mean to account for the intraplot variability.

To reduce the number of parameters to be estimated, the authors assume that the random effects are independent.

Stage 2 specification of the model defined by (13) becomes

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \tag{15}$$

where

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}), \text{ diagonal } \boldsymbol{\Sigma} = \begin{bmatrix} \sigma^2_{u_1} & 0 & 0 \\ 0 & \sigma^2_{u_2} & 0 \\ 0 & 0 & \sigma^2_{u_3} \end{bmatrix}$$

and

$$\text{Var}(\epsilon_{ij}) = \sigma^2 (\widehat{\texttt{weight}}_{ij})^{2\delta}$$

Parameter σ^2 in the above is no longer an overall error variance σ^2_ϵ but a common multiplier or a (squared) scale parameter.

In `menl`, this type of heteroskedasticity is modeled by specifying `resvariance(power _yhat, noconstant)`. `_yhat` designates that the variance should be modeled as a function of predicted values, $\widehat{\texttt{weight}}_{ij}$. By default, variance function `power` includes a constant, which we suppress by specifying the `noconstant` option.

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb) define(phi3: U3[plot], xb)
> resvariance(power _yhat, noconstant)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -364.0225
Iteration 2:    linearization log likelihood = -364.228352
Iteration 3:    linearization log likelihood = -364.43168
Iteration 4:    linearization log likelihood = -364.383194
Iteration 5:    linearization log likelihood = -364.389644
Iteration 6:    linearization log likelihood = -364.389152
Iteration 7:    linearization log likelihood = -364.389201
Iteration 8:    linearization log likelihood = -364.389204

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48

                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10
```

```
Linearization log likelihood =  -364.3892
        phi1:  U1[plot], xb
        phi2:  U2[plot], xb
        phi3:  U3[plot], xb
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	16.82289	.6030526	27.90	0.000	15.64093	18.00485
phi2						
_cons	51.74669	.4579629	112.99	0.000	50.8491	52.64428
phi3						
_cons	7.54537	.0856321	88.11	0.000	7.377534	7.713206

Random-effects Parameters		Estimate	Std. Err.	[95% Conf.	Interval]
plot: Independent					
	var(U1)	11.32134	2.831139	6.934848	18.48241
	var(U2)	2.68911	.9344037	1.36093	5.31351
	var(U3)	5.44e-12	.	.	.
Residual variance: Power _yhat					
	sigma2	.0509223	.004422	.0429527	.0603706
	delta	.9339856	.0244477	.886069	.9819023

The near-zero estimate of the variance component of u_{3j}, `var(U3)`, suggests that the random-effects model is overparameterized. The within-group heteroskedasticity structure appears to explain enough variability in our data, and we no longer need random effects specific to ϕ_{3j}. This is quite common in mixed-effects models: the random-effects covariance structure and the within-group error covariance structure compete with each other, in the sense that fewer random effects are needed when the within-group error covariance structure is present, and vice versa.

Let's omit u_{3j} from (15) but now assume an unstructured covariance matrix for u_{1j} and u_{2j}. The EM algorithm used by `menl` to obtain initial values produces the starting values for variance components that are, in general, close to the final estimates upon convergence. Thus it can be used as a tool to help us detect potential convergence problems because of an overparameterized random-effects structure at an earlier stage. For example, we can check whether an unstructured covariance matrix is a reasonable choice for the random effects u_{1j} and u_{2j} for these data by displaying estimates after a few iterations. This can be done by specifying the `iterate(#)` option, where # is a small number of iterations, say, between 1 and 4. Below we specify `iterate(3)` to perform only three iterations and the `stddeviations` option to obtain standard deviations and correlations instead of variances and covariances for easier interpretability:

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb)
> covariance(U*, unstructured) resvariance(power _yhat, noconstant)
> iterate(3) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -379.663433
Iteration 2:    linearization log likelihood = -362.909205
Iteration 3:    linearization log likelihood = -361.923807

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48

                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10

Linearization log likelihood = -361.94127

        phi1:  U1[plot], xb
        phi2:  U2[plot], xb
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	16.92772	.5676773	29.82	0.000	15.81509	18.04035
phi2						
_cons	51.81715	.4484055	115.56	0.000	50.93829	52.69601
/phi3	7.54089	.0869002	86.78	0.000	7.370569	7.711211

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
plot: Unstructured				
sd(U1)	2.904992	.4071122	2.207271	3.823263
sd(U2)	1.282251	.255516	.8676664	1.89493
corr(U1,U2)	-.99999	.0033707	-1	1
Residual variance:				
Power _yhat				
sigma	.225488	.0095078	.2076023	.2449146
delta	.955391	.0230645	.9101854	1.000597

```
Warning: convergence not achieved
```

The `U*` in `covariance(U*, unstructured)` is a shorthand notation to reference all random effects starting with `U`, that is, `U1` and `U2` in this example. The correlation between u_{1j} and u_{2j} is near -1 with a 95% CI of $[-1, 1]$, which indicates that the random-effects model may still be overparameterized. If you try to fit this model without the `iteration(3)` option, it would keep iterating without convergence.

Therefore, we further simplify the random-effects covariance structure by assuming independence between u_{1j} and u_{2j}. Stage 2 specification of the model defined by (13) is now

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 + u_{1j} \\ \beta_2 + u_{2j} \\ \beta_3 \end{bmatrix} \tag{16}$$

where

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}), \text{ diagonal } \mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{u_1} & 0 \\ 0 & \sigma^2_{u_2} \end{bmatrix}$$

and

$$\text{Var}(\epsilon_{ij}) = \sigma^2(\widehat{\texttt{weight}}_{ij})^{2\delta}$$

We fit this model and store its results as S2:

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb)
> resvariance(power _yhat, noconstant)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -402.761818
Iteration 2:    linearization log likelihood = -372.425639
Iteration 3:    linearization log likelihood = -364.009461
Iteration 4:    linearization log likelihood = -364.371515
Iteration 5:    linearization log likelihood = -364.38263
Iteration 6:    linearization log likelihood = -364.38966
Iteration 7:    linearization log likelihood = -364.389167
Iteration 8:    linearization log likelihood = -364.389203

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10

Linearization log likelihood =  -364.3892

        phi1:  U1[plot], xb
        phi2:  U2[plot], xb
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	16.82289	.6030523	27.90	0.000	15.64093	18.00485
phi2						
_cons	51.74669	.4579626	112.99	0.000	50.8491	52.64428
/phi3	7.545369	.085632	88.11	0.000	7.377534	7.713205

Random-effects Parameters		Estimate	Std. Err.	[95% Conf.	Interval]
plot: Independent					
	var(U1)	11.32134	2.831139	6.934847	18.48241
	var(U2)	2.689111	.9344037	1.36093	5.31351
Residual variance:					
Power _yhat					
	sigma2	.0509223	.004422	.0429527	.0603706
	delta	.9339856	.0244477	.886069	.9819023

```
. estimates store S2
```

Because (16) is not nested in (14), we assess the adequacy of the heteroskedastic model by using information criteria. We use `estimates stats` to display the AIC and BIC values for the three models.

```
. estimates stats S1 S2
Akaike's information criterion and Bayesian information criterion
```

Model	Obs	ll(null)	ll(model)	df	AIC	BIC
S1	412	.	-739.8344	10	1499.669	1539.879
S2	412	.	-364.3892	7	742.7784	770.9256

```
Note: N=Obs used in calculating BIC; see [R] BIC note.
```

The heteroskedastic model defined by (16) has smaller AIC and BIC values and thus provides a much better representation of the data than (14).

◁

▷ Example 12: Heteroskedastic model with interactions

The main goal of the soybean study was to compare growth patterns of the two genotypes of soybean over the three growing seasons, represented by calendar years 1988 through 1990. More specifically, we would like to compare the limiting growth, the half-life, and the growth rate of soybeans across growing seasons and genotypes.

Let $P_j = I\left(\texttt{variety}_j = \texttt{P}\right)$ be the indicator for genotype variety P, $S_{89,j} = I\left(\texttt{year}_j = 1989\right)$ be the indicator for growing season 1989, and $S_{90,j} = I\left(\texttt{year}_j = 1990\right)$ be the indicator for growing season 1990. Genotype variety F and growing season 1988 are baselines.

Consider an extension of the model defined by (13) and (16), where, in addition to random effects, ϕ_{1j} includes main and interaction effects of growing seasons and genotype variety, ϕ_{2j} includes main effects of growing seasons and genotype variety, and ϕ_{3j} contains main effects of growing seasons only.

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_{11} + \beta_{12}S_{89,j} + \beta_{13}S_{90,j} + \beta_{14}P_j + \beta_{15}S_{89,j} \times P_j + \beta_{16}S_{90,j} \times P_j + u_{1j} \\ \beta_{21} + \beta_{22}S_{89,j} + \beta_{23}S_{90,j} + \beta_{24}P_j + u_{2j} \\ \beta_{31} + \beta_{32}S_{89,j} + \beta_{33}S_{90,j} \end{bmatrix} \tag{17}$$

To fit the model defined by (13) and (17) by using `menl`, we extend `menl`'s specification from example 11 by including the full-factorial interaction `i.year##i.variety` in the expression `{phi1:}`, main effects `i.year` and `i.variety` in the expression `{phi2:}`, and main effects `i.year` in the expression `{phi3:}`.

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: i.year##i.variety U1[plot])
> define(phi2: i.year i.variety U2[plot])
> define(phi3: i.year, xb)
> resvariance(power _yhat, noconstant)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -292.62615
Iteration 2:    linearization log likelihood = -290.243889
  (iteration log omitted)
Iteration 10:   linearization log likelihood = -290.90729
Iteration 11:   linearization log likelihood = -290.907297
```

```
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10
                                                Wald chi2(10)     =     413.88
Linearization log likelihood =  -290.9073       Prob > chi2       =     0.0000
```

phi1: i.year i.variety i.year#i.variety U1[plot]
phi2: i.year i.variety U2[plot]
phi3: i.year

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
year						
1989	-8.837933	1.056113	-8.37	0.000	-10.90788	-6.76799
1990	-3.666206	1.165969	-3.14	0.002	-5.951463	-1.380949
variety						
P	1.648139	1.033433	1.59	0.111	-.3773533	3.673631
year#variety						
1989#P	5.563008	1.167783	4.76	0.000	3.274196	7.851819
1990#P	.0974816	1.178054	0.08	0.934	-2.211462	2.406425
_cons	19.42734	.9445749	20.57	0.000	17.57601	21.27867
phi2						
year						
1989	-2.253227	.9746496	-2.31	0.021	-4.163505	-.3429493
1990	-4.970736	.9778318	-5.08	0.000	-6.887251	-3.054221
variety						
P	-1.294058	.4255317	-3.04	0.002	-2.128085	-.4600314
_cons	54.81257	.758724	72.24	0.000	53.3255	56.29964
phi3						
year						
1989	-.9023768	.1992358	-4.53	0.000	-1.292872	-.5118818
1990	-.6805314	.2100799	-3.24	0.001	-1.092281	-.2687823
_cons	8.060677	.1459662	55.22	0.000	7.774588	8.346765

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
plot: Independent				
var(U1)	.8643051	.5271731	.2615089	2.856589
var(U2)	.1341755	.2306995	.0046146	3.901367
Residual variance:				
Power _yhat				
sigma2	.0467091	.0039176	.0396286	.0550546
delta	.9451193	.0227608	.9005089	.9897297

```
. estimates store S3
```

By including more fixed effects in the model, which explain some of the variability in the average leaf weight, we substantially reduced the estimates of variance components. Compared with example 11, `var(U1)` decreased from 11.32 to 0.86, and `var(U2)` decreased from 2.69 to 0.13. It often happens that specifying a better-fitting model for the fixed effects reduces the need for random effects in the model.

We can compare model `S3` or the model defined by (17) with model `S2` or the one defined by (16) by using, for example, information criteria.

```
. estimates stats S2 S3
Akaike's information criterion and Bayesian information criterion
```

Model	Obs	ll(null)	ll(model)	df	AIC	BIC
S2	412	.	-364.3892	7	742.7784	770.9256
S3	412	.	-290.9073	17	615.8146	684.172

Note: N=Obs used in calculating BIC; see **[R] BIC note**.

Even though `S3` has many more parameters, it fits the soybean data better than `S2`.

By inspecting the fixed-effects estimates from the output of model `S3`, we see that both the type of year and genotype variety affect all three parameters: the expected maximum leaf weight, half-life, and scale. For example, all three parameters achieve their highest values in the dry year, baseline year 1988, because coefficient estimates for the other years are negative. Also, the genotype variety P reaches its half-life roughly a day later ($\beta_{24} = -1.29$) than genotype variety F.

◁

▷ Example 13: Obtaining predictions

After estimation, we may want to obtain predicted values for the outcome or for the parameters of interest. Continuing with example 12, we want to predict the asymptotic average leaf weight per soybean plant in each plot, $\widehat{\phi_{1j}}$. The ϕ_{1j} parameter is not constant but varies for each plot, growing season, and genotype variety. We can use `predict` after `menl` to obtain predicted values for ϕ_{1j}; see [ME] **menl postestimation**.

First, we create a new grouping variable for growing seasons, genotype variety, and plot types. We also create the `tolist` variable to mark the first observation in each group.

```
. egen group = group(year variety plot)
. by group, sort: generate byte tolist=(_n==1)
```

Next, we use `predict` to compute predicted values for the expression `{phi1:}` and store them in the new variable `phi1`. We store only unique values in `phi1`, one for each `group`; the remaining observations are replaced with missing values.

```
. predict double (phi1 = {phi1:})
. qui replace phi1 = . if tolist!=1
```

We now list the five smallest and the five largest values of the asymptotic average leaf weight.

```
. sort phi1
. list plot year variety phi1 if (_n<=5 | _n>43) & phi1<., sep(5)
```

	plot	year	variety	phi1
1.	1989F6	1989	F	8.8421451
2.	1989F4	1989	F	10.449521
3.	1989F5	1989	F	10.473849
4.	1989F1	1989	F	10.721364
5.	1989F7	1989	F	10.810197
44.	1988P8	1988	P	20.86739
45.	1988P2	1988	P	21.237691
46.	1988P4	1988	P	21.310511
47.	1988P3	1988	P	21.506007
48.	1988P6	1988	P	21.581873

Soybean plants with genotype variety P have substantially larger asymptotic average leaf weight in the dry year, 1988, than soybean plants with genotype variety F in the wet year, 1989.

◁

▷ Example 14: Within-group error correlation structure

Pinheiro and Bates (2000, chap. 8) analyzed data from a study of the estrus cycles of mares. Originally analyzed in Pierson and Ginther (1987), the data contain daily records of the number of ovarian follicles larger than 10 mm over a period ranging from 3 days before ovulation to 3 days after the subsequent ovulation. The measurement times for each mare are scaled so that the ovulations for each mare occur at times 0 and 1 and are recorded in `stime`.

The considered model is

$$\texttt{follicles}_{ij} = \phi_{1j} + \phi_{2j}\sin\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \phi_{4j}\cos\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

where ϕ_{1j} is an intercept, ϕ_{3j} is the frequency of the sine wave for the jth mare, and ϕ_{2j} and ϕ_{4j} are terms determining the amplitude and phase of the sine wave for the jth mare. If a_j and p_j are the amplitude and phase for mare j, then $\phi_{2j} = a_j\cos(p_j)$ and $\phi_{4j} = a_j\sin(p_j)$.

This model was fit in example 8 of [ME] **mixed** in the context of a linear mixed-effects model, where the number of ovarian follicles was a periodic function of time with known frequency ϕ_{3j} equal to 1. If we want to estimate frequency, we cannot use the `mixed` command, because ϕ_{3j} enters the model nonlinearly.

Pinheiro and Bates (2000) suggested an AR(1) correlation structure for modeling the within-group error correlation. This structure can be specified by using the `rescorrelation()` option as `rescorrelation(ar 1, t(time))`, where `time` is an integer-valued time variable used to order the observations within mares and to determine the lags between successive observations.

We also considered several random-effects structures and found that we need only one random intercept to model ϕ_{1j}.

The full specification for the stage 2 model is

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \\ \phi_{4j} \end{bmatrix} = \begin{bmatrix} \beta_1 + u_{1j} \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

where

$$\mathbf{u}_j = u_{1j} \sim N\left(0, \sigma_u^2\right),\ \boldsymbol{\epsilon}_j \sim N(\mathbf{0},\ \sigma_\epsilon^2 \boldsymbol{\Lambda}_j)$$

and

$$\sigma_\epsilon^2 \boldsymbol{\Lambda}_j = \sigma_\epsilon^2 \begin{bmatrix} 1 & \rho & \rho^2 & \dots & \rho^{n_j-1} \\ \rho & 1 & \rho & \dots & \rho^{n_j-2} \\ \rho^2 & \rho & 1 & \dots & \rho^{n_j-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n_j-1} & \rho^{n_j-2} & \rho^{n_j-3} & \dots & 1 \end{bmatrix}$$

We fit this model by using `menl` as follows:

```
. use http://www.stata-press.com/data/r15/ovary, clear
(Ovarian follicles in mares)
. menl follicles = {phi1: U1[mare], xb} + {phi2}*sin(2*_pi*stime*{phi3}) +
> {phi4}*cos(2*_pi*stime*{phi3}), rescorrelation(ar 1, t(time))
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -789.434153
Iteration 2:    linearization log likelihood = -789.434391
Iteration 3:    linearization log likelihood = -789.434391
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
Linearization log likelihood = -789.43439
        phi1:  U1[mare], xb
```

follicles	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	11.98929	.9055946	13.24	0.000	10.21436	13.76422
/phi2	.2226033	.3290159	0.68	0.499	-.4222559	.8674626
/phi3	4.18747	.2746499	15.25	0.000	3.649166	4.725774
/phi4	.279653	.3223277	0.87	0.386	-.3520977	.9114036

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
mare: Identity				
var(U1)	4.935352	3.967836	1.020903	23.85898
Residual: AR(1), time time				
var(e)	20.14587	3.492937	14.34177	28.29888
corr	.7332304	.0463231	.6287332	.8117157

By using estimates of ϕ_{2j} and ϕ_{4j}, we can compute the amplitude and phase for the sine wave for mare j. The amplitude and the phase are the same for all the mares because ϕ_{2j} and ϕ_{4j} are constant and not mare specific.

For example, the amplitude a_j can be computed as $\sqrt{\phi_{2j}^2 + \phi_{4j}^2}$ by using the relationship $\phi_{2j}^2 + \phi_{4j}^2 = a_j^2 \left\{\sin^2(p_j) + \cos^2(p_j)\right\} = a_j^2$. The phase p_j can be computed as $p_j = \text{atan}(\phi_{4j}/\phi_{2j})$ by using the relationship $\phi_{4j}/\phi_{2j} = \{a_j \sin(p_j)\} / \{a_j \cos(p_j)\} = \tan(p_j)$.

We can use `nlcom` to compute the amplitude and the phase.

```
. nlcom (amplitude: sqrt(_b[/phi2]^2 + _b[/phi4]^2))
> (phase: atan(_b[/phi4]/_b[/phi2]))
   amplitude:  sqrt(_b[/phi2]^2 + _b[/phi4]^2)
       phase:  atan(_b[/phi4]/_b[/phi2])
```

follicles	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
amplitude	.3574325	.2451183	1.46	0.145	-.1229904	.8378555
phase	.8985001	1.090985	0.82	0.410	-1.23979	3.03679

As we mentioned in example 1, it is important to try different initial values when fitting NLME models to investigate potential convergence to a local maximum, especially for models containing periodic functions, as in our example. We explore different initial values for this model in *Linearization approach to finding initial values* by considering the functional form of the mean function and arrive at a different solution with a larger log likelihood.

◁

Restricted maximum likelihood

Like `mixed`, `menl` provides estimation by using ML or REML. The difference between the two approaches is described in detail in *Likelihood versus restricted likelihood* in [ME] **mixed**. Briefly, REML is preferable when you have a small number of groups because it produces unbiased, at least for balanced data, estimates of variance components. In large samples, there is little difference between ML and REML. One disadvantage of REML, however, is that LR tests based on REML are inappropriate for comparing models with different fixed-effects specifications.

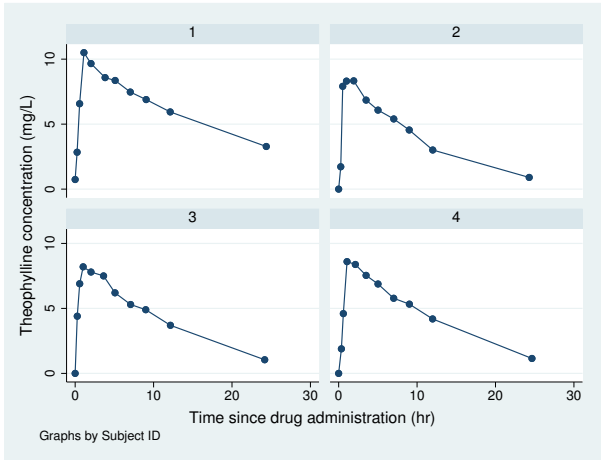
▷ Example 15: Pharmacokinetics modeling

Pharmacokinetics (PKs) is the study of drug absorption, distribution, metabolism, and excretion. It is often referred to as the study of "what the body does with a drug". The goal of PK modeling is to summarize the concentration-time measurements using a model that relates drug input to drug

response, to relate the parameters of this model to patient characteristics, and to provide individual dose-response predictions to optimize individual doses. In other words, by understanding between-subject variation in drug disposition, we can individualize the dosage regimen for a particular patient based on relevant physiological information identified by our PK model.

Consider a PK study of the antiasthmatic agent theophylline that was reported by Boeckmann, Sheiner, and Beal ([1994] 2011) and analyzed by Davidian and Giltinan (1995). The drug was administrated orally to 12 subjects, where dosage `dose` (mg/kg) was given on a per weight basis. Serum concentrations (in mg/L) were obtained at 11 time points per subject over 25 hours following administration. The graph below shows the resulting concentration-time profiles for four subjects.

```
. use http://www.stata-press.com/data/r15/theoph
(Theophylline kinetics (Boeckmann et al., 1994))
. twoway connected conc time if subject<=4, connect(L) by(subject)
```



In PKs, the pattern of rapid rise to a peak concentration followed by an apparent exponential decay may be described by a so-called one-compartment open model with first-order absorption and elimination. The model corresponds roughly to viewing the body as one "blood compartment" and is particularly useful for the PK analysis of drugs that distribute relatively rapidly throughout the body, which makes it a reasonable model for the kinetics of theophylline after oral administration. Further details about compartmental modeling may be found in Gibaldi and Perrier (1982). The one-compartment open model for theophylline kinetics may be expressed as

$$\texttt{conc}_{ij} = \frac{\texttt{dose}_j k_{e_j} k_{a_j}}{\mathrm{Cl}_j \left(k_{a_j} - k_{e_j}\right)} \left\{ \exp\left(-k_{e_j}\texttt{time}_{ij}\right) - \exp\left(-k_{a_j}\texttt{time}_{ij}\right) \right\} + \epsilon_{ij} \tag{18}$$

for $i = 1, \ldots, 11$ and $j = 1, \ldots, 12$. Model parameters are the elimination rate constant k_{e_j}, the absorption rate constant k_{a_j}, and the clearance Cl_j for each subject j.

Because each of the model parameters must be positive to be meaningful, we write

$$\begin{aligned} \mathrm{Cl}_j &= \exp\left(\beta_0 + u_{0j}\right) \\ k_{a_j} &= \exp\left(\beta_1 + u_{1j}\right) \\ k_{e_j} &= \exp\left(\beta_2\right) \end{aligned}$$

where u_{0j} and u_{1j} are assumed independent and normally distributed with means zero and variance $\sigma^2_{u_0}$ and $\sigma^2_{u_1}$, respectively.

The model defined by (18) implies that the predicted value for the concentration at time $\texttt{time}_{ij} = 0$ is $\widehat{\texttt{conc}}_{ij} = 0$. Therefore, a power variance function, a natural candidate for this type of heteroskedastic pattern, cannot be used in this example because error variance will be 0 at $\texttt{time}_{ij} = 0$. So the constant plus power variance function, which adds a constant to the power term, is used instead to model the within-group error variance:

$$\text{Var}(\epsilon_{ij}) = \sigma^2\{(\widehat{\texttt{conc}}_{ij})^{\delta} + c\}^2$$

In `menl`, we use the `resvariance(power _yhat)` option to specify the constant plus power variance function and the following model specification:

```
. menl conc = (dose*{ke:}*{ka:}/({cl:}*({ka:}-{ke:})))*
> (exp(-{ke:}*time)-exp(-{ka:}*time)), define(cl: exp({b0}+{U0[subject]}))
> define(ka: exp({b1}+{U1[subject]})) define(ke: exp({b2}))
> resvariance(power _yhat)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -167.519533
Iteration 2:    linearization log likelihood = -167.65729
  (iteration log omitted)
Iteration 26:   linearization log likelihood = -167.679657
Iteration 27:   linearization log likelihood = -167.679641

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        132
Group variable: subject                         Number of groups  =         12
                                                Obs per group:
                                                              min =         11
                                                              avg =       11.0
                                                              max =         11

Linearization log likelihood = -167.67964

         cl:  exp({b0}+{U0[subject]})
         ka:  exp({b1}+{U1[subject]})
         ke:  exp({b2})
```

conc	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b0	-3.227479	.0598389	-53.94	0.000	-3.344761	-3.110197
/b1	.432931	.1980835	2.19	0.029	.0446945	.8211674
/b2	-2.453742	.0514567	-47.69	0.000	-2.554595	-2.352889

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
subject: Independent				
var(U0)	.0288787	.0127763	.0121337	.0687323
var(U1)	.4075667	.1948713	.1596654	1.040367
Residual variance:				
Power _yhat				
sigma2	.0976905	.0833027	.018366	.519624
delta	.3187133	.2469511	-.1653019	.8027285
_cons	.7288982	.3822952	.2607507	2.03755

The number of groups, 12, is fairly small in these data, so we now refit the model by using REML estimation.

```
. menl conc = (dose*{ke:}*{ka:}/({cl:}*({ka:}-{ke:})))*
> (exp(-{ke:}*time)-exp(-{ka:}*time)), define(cl: exp({b0}+{U0[subject]}))
> define(ka: exp({b1}+{U1[subject]})) define(ke: exp({b2}))
> resvariance(power _yhat) reml

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log restricted-likelihood = -172.31734
Iteration 2:    linearization log restricted-likelihood = -172.423246
  (iteration log omitted)
Iteration 23:   linearization log restricted-likelihood = -172.443827
Iteration 24:   linearization log restricted-likelihood = -172.443845

Computing standard errors:

Mixed-effects REML nonlinear regression         Number of obs     =        132
Group variable: subject                         Number of groups  =         12

                                                Obs per group:
                                                              min =         11
                                                              avg =       11.0
                                                              max =         11

Linear. log restricted-likelihood = -172.44384

          cl:  exp({b0}+{U0[subject]})
          ka:  exp({b1}+{U1[subject]})
          ke:  exp({b2})
```

conc	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/b0	-3.227295	.0619113	-52.13	0.000	-3.348639	-3.105951
/b1	.4354519	.2072387	2.10	0.036	.0292716	.8416322
/b2	-2.453743	.0517991	-47.37	0.000	-2.555267	-2.352218

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
subject: Independent				
var(U0)	.0316416	.014531	.0128634	.0778326
var(U1)	.4500585	.2228206	.1705476	1.187661
Residual variance: Power _yhat				
sigma2	.1015759	.086535	.0191263	.5394491
delta	.3106636	.2466547	-.1727707	.7940979
_cons	.7150935	.3745256	.2561837	1.996063

As expected, the estimates of the random-effects variances are slightly larger than the corresponding ML estimates, but we arrive at similar inferential conclusions based on our REML estimates.

◁

▷ Example 16: Nonlinear functions of parameters

A distinctive feature of example 15 is that parameters of interest are nonlinear functions of the estimated parameters and random effects. To interpret parameters that depend on random effects, we can either integrate random effects out of the parameter expression or condition on them. The former parameter estimates are often referred to as population-based estimates. The latter parameter estimates

are referred to as conditional estimates and, when conditioning on zero random effects, $\mathbf{u}_j = 0$, as estimates for an "average" or typical subject. For linear functions, the population-based estimates coincide with the conditional estimates. This is no longer true for nonlinear functions.

In PK modeling, the parameters of interest are clearance, elimination rate, and absorption rate. These are nonlinear functions of the estimated parameters β_0, β_1, β_2 and subject-specific random effects. Depending on the context, we may be interested in their population-based estimates or in their conditional estimates.

In general, obtaining population-based estimates would require numerical integration to integrate the subject-specific random effects out of the expression. In our example, we can compute population-based estimates directly by using the fact that $\exp(u_{0j})$'s and $\exp(u_{1j})$'s are lognormally distributed.

Thus the population-based clearance Cl^P can be computed as $E(\text{Cl}_j) = E\{\exp(\beta_0 + u_{0j})\} = \exp\left(\beta_0 + \sigma^2_{u_0}/2\right)$ and the population-based absorption rate k_a^P as $E\{\exp(\beta_1 + u_{1j})\} = \exp\left(\beta_1 + \sigma^2_{u_1}/2\right)$. The elimination rate k_e does not depend on subject-specific effects and can thus be computed simply as $k_e^P = k_e = \exp(\beta_2)$.

Alternatively, if we want parameters to represent a typical subject, we can simply set $u_{0j} = 0$ and $u_{1j} = 0$ in their expressions. Thus we can compute clearance and absorption rate for a typical subject simply as $\text{Cl} = \exp(\beta_0)$ and $k_a = \exp(\beta_1)$. These formulas can also be viewed as a result of exponentiating population-based log-clearance and log-absorption rate; that is, $\text{Cl} = \exp[E\{\log(\text{Cl}_j)\}] = \exp(\beta_0)$ and $k_a = \exp\left[E\left\{\log(k_{a_j})\right\}\right] = \exp(\beta_1)$.

If we compare the formulas for, say, Cl^P and Cl, the former considers variation in clearances across subjects, whereas the latter ignores such variation and instead reflects what the clearance would be for a typical subject with $u_{0j} = 0$.

Both approaches have merit, and here we will compute, for example, $\text{Cl}^P = \exp(\widehat{\beta}_0 + \widehat{\sigma}^2_{u_0}/2) = \exp(-3.23 + 0.032/2) = 0.04$. That is, 0.04 liters of serum concentration are cleared of the theophylline drug per hour per kg body weight in the considered population. In other words, for the population of subjects that weigh 75 kg, an average of $75 \times 0.04 \approx 3$ liters of serum concentration are cleared of theophylline every hour.

We can also use `nlcom` to compute the estimates of Cl^P and Cl. To use `nlcom`, we need to know how parameters are labeled by `menl` for postestimation. We can use `menl`'s `coeflegend` option to display parameter names. We also specify `noheader` to suppress the table header.

```
. menl, coeflegend noheader
```

conc	Coef.	Legend
/b0	-3.227295	_b[/b0]
/b1	.4354519	_b[/b1]
/b2	-2.453743	_b[/b2]
/subject		
lnsd(U0)	-1.726641	_b[/subject:lnsd(U0)]
lnsd(U1)	-.3991888	_b[/subject:lnsd(U1)]
/Residual		
lnsigma	-1.143475	_b[/Residual:lnsigma]
delta	.3106636	_b[/Residual:delta]
ln_cons	-.335342	_b[/Residual:ln_cons]

If we examine the output carefully, we will notice that `menl, coeflegend` displayed results in the estimation metric—as log-standard deviations instead of variances. Although by default `menl` displays

parameters in their original metric, it stores them in the estimation metric, the metric that was used during optimization; see *Examples of specifying initial values* and *Methods and formulas* for more details about the estimation metric.

The parameters we need to compute Cl^P and Cl are coefficient `_b[/b0]` and the variance of `U0`, which can be obtained as `exp(2*_b[/subject:lnsd(U0)])` based on the stored estimate of the log-standard deviation of `U0`. We now use `nlcom` to compute our nonlinear estimates.

```
. nlcom (Cl_P: exp(_b[/b0]+0.5*exp(2*_b[/subject:lnsd(U0)]))) (Cl: exp(_b[/b0]))
       Cl_P:  exp(_b[/b0]+0.5*exp(2*_b[/subject:lnsd(U0)]))
         Cl:  exp(_b[/b0])
```

conc	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Cl_P	.0402972	.002512	16.04	0.000	.0353738	.0452205
Cl	.0396646	.0024557	16.15	0.000	.0348516	.0444777

Working with parameters in the estimation metric can be tedious, especially when nonlinear expressions contain multiple variance components. In that case, you may consider using `estat sd` after `menl` to obtain results in the standard deviation metric or, if you also specify the `variance` option, in the variance metric; see [ME] **menl postestimation**. If you specify the `post` option with `estat sd`, the results will also be stored in the standard deviation or variance metrics, which you can use for further postestimation analysis.

```
. estat sd, post variance coeflegend
```

conc	Coef.	Legend
/b0	-3.227295	_b[/b0]
/b1	.4354519	_b[/b1]
/b2	-2.453743	_b[/b2]

Random-effects Parameters		Estimate	Legend
subject: Independent			
	var(U0)	.0316416	_b[/subject:var(U0)]
	var(U1)	.4500585	_b[/subject:var(U1)]
Residual variance:			
Power _yhat			
	sigma2	.1015759	_b[/Residual:sigma2]
	delta	.3106636	_b[/Residual:delta]
	_cons	.7150935	_b[/Residual:_cons]

In addition to results being displayed in the variance metric, because of the `post` option, they are stored in that metric. We also specified the `coeflegend` option with `estat sd` to see how parameters are labeled so that we could refer to them in other postestimation commands such as `nlcom`.

Now, we can simply refer to the variance of `U0` as `_b[/subject:var(U0)]` in our `nlcom` command.

```
. nlcom (Cl_P: exp(_b[/b0]+0.5*_b[/subject:var(U0)]))
       Cl_P:  exp(_b[/b0]+0.5*_b[/subject:var(U0)])
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Cl_P	.0402972	.002512	16.04	0.000	.0353738	.0452205

`estat sd`'s `post` option should be used with caution because it clears all estimation results except the parameter estimates in `e(b)` and their VCE in `e(V)`. Thus the only postestimation features that will work after `estat sd, post` are those that need only `e(b)` and `e(V)`, such as `lincom` and `nlcom`. Other postestimation features will not be available, and you will need to refit your model to run them. To avoid refitting your model, you may consider storing your estimation results in memory (see [R] **estimates store**) or saving them on disk (see [R] **estimates save**) before using `estat sd, post`. We no longer needed the estimation results from `menl`, so we did not mind clearing them.

◁

Three-level models

Representation of (1) can be extended to, for example, two-nested levels of clustering, to form the following three-level model, with observations composing the first level,

$$\mathbf{y}_{jk} = \boldsymbol{\mu}\left(\mathbf{X}_{jk}, \boldsymbol{\beta}, \mathbf{u}_k^{(3)}, \mathbf{u}_{jk}^{(2)}\right) + \boldsymbol{\epsilon}_{jk}$$

where the first-level observations $i = 1, \ldots, n_{jk}$ are nested within the second-level groups $j = 1, \ldots, M_k$, which are nested within the third-level groups $k = 1, \ldots, M$. Group j nested within group k consists of n_{jk} observations, so $\mathbf{y}_{jk}$, $\mathbf{X}_{jk}$, and $\boldsymbol{\epsilon}_{jk}$ each have row dimension n_{jk}.

Also, assume that

$$\mathbf{u}_k^{(3)} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_3) \qquad \mathbf{u}_{jk}^{(2)} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_2) \qquad \boldsymbol{\epsilon}_{jk} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{\Lambda}_{jk})$$

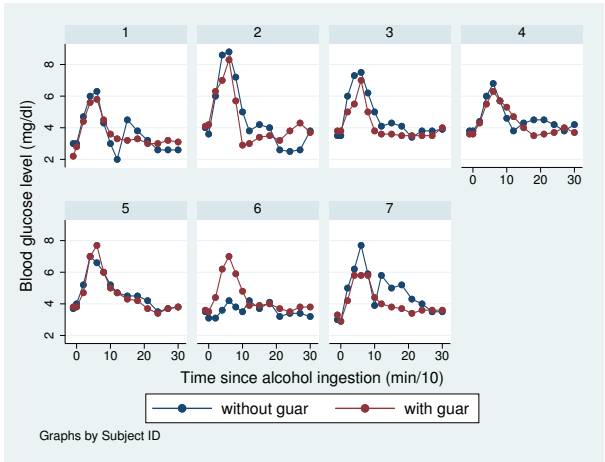
and that $\mathbf{u}_k^{(3)}$, $\mathbf{u}_{jk}^{(2)}$, and $\boldsymbol{\epsilon}_{jk}$ are independent.

▷ Example 17: Three-level model

Hand and Crowder (1996, 118–120) analyzed a study where the blood glucose levels `glucose` of 7 volunteers, `subject`, who took alcohol at time 0 were measured 14 times, `time`, over a period of 5 hours after alcohol consumption. The same experiment was repeated at a later date with the same subjects but with a dietary additive, guar, used for all subjects. Variable `guar` is a binary variable that identifies whether a subject received a dietary additive. It also identifies each experiment, with 0 corresponding to the experiment without guar and 1 corresponding to the experiment with guar. Thus we will use the `guar` variable both as the level indicator and, later, as a fixed-effects variable.

Here is a plot of the whole dataset.

```
. use http://www.stata-press.com/data/r15/glucose
(Glucose levels following alcohol ingestion (Hand and Crowder, 1996))
. twoway connected glucose time if guar==0 ||
> connected glucose time if guar==1 ||, by(subject, rows(2))
> legend(order(1 "without guar" 2 "with guar"))
```



Our preliminary assessment based on the above graph is that, except for subject 6, the effect of the dietary additive guar on the temporal trajectory of the blood glucose levels does not seem to be important. The effect of guar will be formally tested in example 18.

Hand and Crowder (1996) proposed the following empirical model relating the expected glucose level to time,

$$\texttt{glucose}_{ijk} = \phi_{1jk} + \phi_{2jk}\texttt{time}^3 \exp\left(-\phi_{3jk}\texttt{time}\right) + \epsilon_{ijk} \tag{19}$$

where $k = 1, \ldots, 7$, $j = 1, 2$, and $i = 1, \ldots, 14$. The blood glucose level is ϕ_1 at $\texttt{time} = 0$ and as $\texttt{time} \to \infty$. This is intentional, so that ϕ_1 can be interpreted as both the blood glucose level before ingesting alcohol and the blood glucose level after the effect of alcohol ingestion has washed out.

Pinheiro and Bates (2000, exercise 3, 412) analyzed this dataset in the context of a three-level NLME model. They initially proposed the following stage 2 specification,

$$\begin{aligned} \phi_{1jk} &= \beta_1 + u^{(3)}_{1k} + u^{(2)}_{1j,k} \\ \phi_{2jk} &= \beta_2 + u^{(3)}_{2k} + u^{(2)}_{2j,k} \\ \phi_{3jk} &= \beta_3 \end{aligned} \tag{20}$$

$$\mathbf{u}^{(3)}_k = \begin{bmatrix} u^{(3)}_{1k} \\ u^{(3)}_{2k} \end{bmatrix} \sim N\left(\mathbf{0}, \boldsymbol{\Sigma}_3\right) \qquad \mathbf{u}^{(2)}_{j,k} = \begin{bmatrix} u^{(2)}_{1j,k} \\ u^{(2)}_{2j,k} \end{bmatrix} \sim N\left(\mathbf{0}, \boldsymbol{\Sigma}_2\right) \qquad \epsilon_{ijk} \sim N\left(0, \sigma^2_\epsilon\right)$$

where $\boldsymbol{\Sigma}_2$ and $\boldsymbol{\Sigma}_3$ are general symmetric covariance matrices. $u^{(2)}_{1j,k}$ and $u^{(2)}_{2j,k}$ are random intercepts at the guar-within-subject level and can be specified in menl as `UU1[subject>guar]` and `UU2[subject>guar]`.

The full model defined by (19) and (20) contains many parameters. We will follow our own advice from example 11 and specify the `iterate()` option to check how reasonable our model is for the data we have.

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3}*time),
> define(phi1: U1[subject] UU1[subject>guar])
> define(phi2: U2[subject] UU2[subject>guar])
> covariance(U1 U2, unstructured) covariance(UU*, unstructured)
> stddeviations iterate(3)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -189.447112
Iteration 2:    linearization log likelihood = -189.441164
Iteration 3:    linearization log likelihood = -189.441123
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        196
```

Path	No. of Groups	Observations per Group Minimum	Average	Maximum
subject	7	28	28.0	28
subject>guar	14	14	14.0	14

```
Linearization log likelihood = -189.44112
       phi1:  U1[subject] UU1[subject>guar]
       phi2:  U2[subject] UU2[subject>guar]
```

glucose	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	3.661565	.1160345	31.56	0.000	3.434142	3.888989
phi2						
_cons	.4283298	.0530028	8.08	0.000	.3244462	.5322134
/phi3	.5896813	.013861	42.54	0.000	.5625144	.6168483

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
subject: Unstructured				
sd(U1)	.2624559	.0926845	.1313592	.5243873
sd(U2)	.0598426	.0724586	.0055765	.6421878
corr(U1,U2)	-.1489676	.9201413	-.9636339	.9346912
subject>guar: Unstructured				
sd(UU1)	.0919531	.0764232	.0180353	.468823
sd(UU2)	.1227072	.0412876	.0634547	.2372882
corr(UU1,UU2)	.99999	.0047062	-1	1
sd(Residual)	.5712261	.0305339	.514409	.6343187

```
Warning: convergence not achieved
```

The estimated correlation `corr(UU1,UU2)` is near one with the confidence interval spanning the entire range for the correlation parameter, which indicates that the random-effects structure is overparameterized. The confidence interval for `corr(U1,U2)` contains zero, which suggests that this term does not contribute much to explaining between-subject variability. If we try to fit this model without the `iterate()` option, it will continue iterating without convergence.

We simplify our model by assuming independence between random effects; that is, we assume that random-effects covariance matrices Σ_2 and Σ_3 are diagonal.

Recall that `covariance(, independent)` is assumed by default, so we do not need to explicitly specify the `covariance()` option:

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3}*time),
> define(phi1: U1[subject] UU1[subject>guar])
> define(phi2: U2[subject] UU2[subject>guar]) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -190.355295
Iteration 2:    linearization log likelihood = -190.36034
Iteration 3:    linearization log likelihood = -190.363301
Iteration 4:    linearization log likelihood = -190.364176
Iteration 5:    linearization log likelihood = -190.363753
Iteration 6:    linearization log likelihood = -190.363967
Iteration 7:    linearization log likelihood = -190.363859
Iteration 8:    linearization log likelihood = -190.363914
Iteration 9:    linearization log likelihood = -190.363886

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        196
```

Path	No. of Groups	Observations per Group Minimum	Average	Maximum
subject	7	28	28.0	28
subject>guar	14	14	14.0	14

```
Linearization log likelihood = -190.36389

       phi1:  U1[subject] UU1[subject>guar]
       phi2:  U2[subject] UU2[subject>guar]
```

glucose	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	3.658712	.1168642	31.31	0.000	3.429662	3.887762
phi2						
_cons	.4239173	.0526333	8.05	0.000	.320758	.5270766
/phi3	.5876636	.0137214	42.83	0.000	.5607701	.6145571

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
subject: Independent				
sd(U1)	.2685609	.092104	.137126	.5259757
sd(U2)	.0422075	.1078501	.0002821	6.315554
subject>guar: Independent				
sd(UU1)	.0666034	.1527526	.0007435	5.96621
sd(UU2)	.1362263	.0433548	.0730065	.2541912
sd(Residual)	.5732488	.0309928	.5156118	.6373288

The random-effects structure may still be overparameterized, given small estimates for `sd(U2)` and `sd(UU1)`. If we were to perform an LR test of the corresponding variance components being zero,

we would have no statistical evidence to reject this null hypothesis; see example 7 for an instance of performing an LR test.

◁

▷ Example 18: Three-level model with continuous-time AR(1) error structure

The main objective of the study from example 17 was to determine whether the use of the dietary additive guar significantly affected time profiles of the blood glucose levels of subjects.

We continue with the model without random effects `U2[subject]` and `UU1[subject>guar]` and include covariate `guar` for all ϕ_{jk}'s. Hand and Crowder (1996) also suggested to use a continuous-time AR(1) correlation structure for the `guar`-within-`subject` errors, which is specified in `menl` as `rescorrelation(ctar1, t(time))`:

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3:}*time),
> define(phi1: i.guar U1[subject]) define(phi2: i.guar UU2[subject>guar])
> define(phi3: i.guar, xb) rescorrelation(ctar1, t(time)) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -180.623038
Iteration 2:    linearization log likelihood = -181.270273
  (iteration log omitted)
Iteration 24:   linearization log likelihood = -181.187002
Iteration 25:   linearization log likelihood = -181.186986

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        196
```

Path	No. of Groups	Observations per Group Minimum	Average	Maximum
subject	7	28	28.0	28
subject>guar	14	14	14.0	14

```
                                                Wald chi2(3)      =       0.66
Linearization log likelihood = -181.18699       Prob > chi2       =     0.8814

       phi1:  i.guar U1[subject]
       phi2:  i.guar UU2[subject>guar]
       phi3:  i.guar
```

glucose	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
guar						
with guar	-.0814355	.1532735	-0.53	0.595	-.381846	.218975
_cons	3.685365	.1433368	25.71	0.000	3.40443	3.9663
phi2						
guar						
with guar	.0109469	.0883807	0.12	0.901	-.162276	.1841698
_cons	.344372	.0606914	5.67	0.000	.2254191	.4633248
phi3						
guar						
with guar	.0103743	.0330196	0.31	0.753	-.054343	.0750916
_cons	.5514012	.022009	25.05	0.000	.5082642	.5945381

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
subject: Identity				
sd(U1)	.2453634	.1013232	.1092206	.5512071
subject>guar: Identity				
sd(UU2)	.1011852	.0276419	.0592358	.1728421
Residual: CTAR1, time time				
sd(e)	.6208598	.0412948	.5449771	.7073086
corr	.6547722	.0564848	.5440641	.7654804

The dietary additive guar does not seem to affect the blood-glucose-level profiles over time. This actually conforms with the plot of the data from example 17, where, except for subject 6, the profiles with and without guar are similar.

◁

▷ Example 19: Three-level model with block-diagonal covariance matrix

Pinheiro and Bates (2000) report the data from the experiment conducted by Microelectronics Division of Lucent Technologies to study the variability in the manufacturing of analog MOS circuits. The intensities of the `current` (in mA) were collected on n-channel devices at five ascending `voltages`: 0.8, 1.2, 1.6, 2.0, and 2.4 V. Measurements were made on 8 `sites` of each of 10 `wafers`. The main objective of the study was to build an empirical model to simulate the behavior of similar circuits.

The intensity of the current at the ith level of `voltage` in the jth `site` within the kth `wafer` is expressed as

$$\texttt{current}_{ijk} = \phi_{1jk} + \phi_{2jk}\cos\left(\phi_{3jk}\texttt{voltage}_i + \pi/4\right) + \epsilon_{ijk}$$

where

$$\phi_{1jk} = \beta_0 + u^{(3)}_{0k} + u^{(2)}_{0j,k} + \left(\beta_1 + u^{(3)}_{1k} + u^{(2)}_{1j,k}\right)\texttt{voltage}_i + \left(\beta_2 + u^{(3)}_{2k} + u^{(2)}_{2j,k}\right)\texttt{voltage}_i^2$$
$$\phi_{2jk} = \beta_3 + u^{(3)}_{3k} + u^{(2)}_{3j,k}$$
$$\phi_{3jk} = \beta_4 + u^{(3)}_{4k}$$

$$\mathbf{u}^{(3)}_k = \begin{bmatrix} u^{(3)}_{0k} \\ u^{(3)}_{1k} \\ u^{(3)}_{2k} \\ u^{(3)}_{3k} \\ u^{(3)}_{4k} \end{bmatrix} \sim N\left(\mathbf{0}, \mathbf{\Sigma}_3\right) \qquad \mathbf{u}^{(2)}_{j,k} = \begin{bmatrix} u^{(2)}_{0j,k} \\ u^{(2)}_{1j,k} \\ u^{(2)}_{2j,k} \\ u^{(2)}_{3j,k} \end{bmatrix} \sim N\left(\mathbf{0}, \mathbf{\Sigma}_2\right) \qquad \epsilon_{ijk} \sim N\left(0, \sigma^2_\epsilon\right)$$

Parameters β_0, β_1, and β_2 characterize the quadratic component of the model, and amplitude β_3 and frequency β_4 characterize the periodic component represented by the cosine wave.

For illustration, consider the following random-effects covariance structures:

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} \sigma^{(3)}_{11} & & & & & \\ & \sigma^{(3)}_{22} & & & & \\ & & \sigma^{(3)}_{33} & & & \\ & & & \sigma^{(3)}_{44} & & \\ & & & & \sigma^{(3)}_{55} \end{bmatrix} \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix} \sigma^{(2)}_{11} & \sigma^{(2)}_{12} & 0 & 0 \\ \sigma^{(2)}_{12} & \sigma^{(2)}_{22} & 0 & 0 \\ 0 & 0 & \sigma^{(2)}_{33} & \sigma^{(2)}_{34} \\ 0 & 0 & \sigma^{(2)}_{34} & \sigma^{(2)}_{44} \end{bmatrix}$$

If we were to fit this model by using `menl`, we would type

```
. use http://www.stata-press.com/data/r15/wafer
(Modeling of analog MOS circuits)
. menl current = {phi1:}+{phi2:}*cos({phi3:}*voltage + _pi/4),
> define(phi1: voltage c.voltage#c.voltage W0[wafer] S0[wafer>site]
>              c.voltage#(W1[wafer] S1[wafer>site])
>              c.voltage#c.voltage#(W2[wafer] S2[wafer>site]))
> define(phi2: W3[wafer] S3[wafer>site]) define(phi3: W4[wafer], xb)
> covariance(S0 S1, unstructured) covariance(S2 S3, unstructured)
> covariance(W*, independent) stddeviations
```

In the specification above, $\boldsymbol{\Sigma}_3$ is specified as `covariance(W*, independent)`, although this specification could have been omitted because `independent` is `menl`'s default random-effects covariance structure. The block-diagonal matrix $\boldsymbol{\Sigma}_2$ is specified by using repeated `covariance()` options: `covariance(S0 S1, unstructured)` and `covariance(S2 S3, unstructured)`. If we tried to run this model, we would find out that it is overparameterized.

Because of the large number of random effects at each grouping level, to avoid numerically unstable estimates, we will further simplify our model by assuming independence between $u^{(2)}_{2j,k}$ and $u^{(2)}_{3j,k}$, which implies that $\sigma^{(2)}_{34} = 0$:

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} \sigma^{(3)}_{11} & & & & & \\ & \sigma^{(3)}_{22} & & & & \\ & & \sigma^{(3)}_{33} & & & \\ & & & \sigma^{(3)}_{44} & & \\ & & & & \sigma^{(3)}_{55} \end{bmatrix} \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix} \sigma^{(2)}_{11} & \sigma^{(2)}_{12} & 0 & 0 \\ \sigma^{(2)}_{12} & \sigma^{(2)}_{22} & 0 & 0 \\ 0 & 0 & \sigma^{(2)}_{33} & 0 \\ 0 & 0 & 0 & \sigma^{(2)}_{44} \end{bmatrix}$$

We now try to fit the above simpler model. Note that given the complexity of this model, it takes some time to execute.

```
. use http://www.stata-press.com/data/r15/wafer
(Modeling of analog MOS circuits)
. menl current = {phi1:}+{phi2:}*cos({phi3:}*voltage + _pi/4),
> define(phi1: voltage c.voltage#c.voltage W0[wafer] S0[wafer>site]
>              c.voltage#(W1[wafer] S1[wafer>site])
>              c.voltage#c.voltage#(W2[wafer] S2[wafer>site]))
> define(phi2: W3[wafer] S3[wafer>site]) define(phi3: W4[wafer], xb)
> covariance(S0 S1, unstructured) covariance(S2 S3, independent)
> covariance(W*, independent) stddeviations
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = 737.063827
Iteration 2:    linearization log likelihood = 754.072576
```

```
Iteration 3:    linearization log likelihood = 825.912909
Iteration 4:    linearization log likelihood = 825.917104
Iteration 5:    linearization log likelihood = 825.917107
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        400
```

Path	No. of Groups	Observations per Group Minimum	Average	Maximum
wafer	10	40	40.0	40
wafer>site	80	5	5.0	5

```
                                                Wald chi2(2)      =    8763.93
Linearization log likelihood =  825.91711       Prob > chi2       =     0.0000
       phi1:  voltage c.voltage#c.voltage W0[wafer] S0[wafer>site]
              c.voltage#W1[wafer] c.voltage#S1[wafer>site]
              c.voltage#c.voltage#W2[wafer]
              c.voltage#c.voltage#S2[wafer>site]
       phi2:  W3[wafer] S3[wafer>site]
       phi3:  W4[wafer], xb
```

current	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
voltage	6.046937	.1022632	59.13	0.000	5.846504	6.247369
c.voltage# c.voltage	1.158782	.0159669	72.57	0.000	1.127487	1.190076
_cons	-4.658034	.0361763	-128.76	0.000	-4.728938	-4.58713
phi2						
_cons	.1684428	.002054	82.01	0.000	.1644171	.1724686
phi3						
_cons	6.449391	.0019631	3285.32	0.000	6.445543	6.453238

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
wafer: Independent				
sd(W0)	.1107108	.0262518	.0695589	.1762087
sd(W1)	.3041975	.0764653	.1858624	.4978744
sd(W2)	.0449994	.0125441	.026057	.0777122
sd(W3)	.0057862	.0016144	.0033489	.0099974
sd(W4)	.0061349	.0013878	.0039377	.0095579
wafer>site: Unstructured				
sd(S0)	.0729495	.006297	.0615952	.0863969
sd(S1)	.2930062	.0252425	.2474831	.346903
corr(S0,S1)	-.8113227	.0413367	-.8782248	-.7132762
wafer>site: Independent				
sd(S2)	.0627587	.0053067	.0531738	.0740712
sd(S3)	.0080611	.0006861	.0068227	.0095244
sd(Residual)	.0008407	.0000711	.0007122	.0009922

In this example, our primary focus was to demonstrate how to use `menl` to fit a block-diagonal random-effects covariance structure. But if we were to interpret our fixed-effects estimates, the average frequency of the cosine wave, $\beta_4 = E(\phi_{3jk})$, for example, is estimated to be $6.45V^{-1}$, with a corresponding estimated period of $2\pi/\widehat{\beta}_4 \approx 0.97V$. Also, some of the estimates of standard deviations such as `sd(W2)`, `sd(W3)`, and `sd(W4)` are very small, which suggests that this model may still be too rich for the observed data. If we proceeded to further analyze these data, we would consider simpler models. For example, at the very least, we would have omitted the term `W3[wafer]` from this model.

◁

Obtaining initial values

Obtaining good starting or initial values is important for the estimation of many statistical models, but it is often crucial for the estimation of NLME models. NLME models are known to be sensitive to the initial values and to have difficulty converging. Highly nonlinear mean specification or complicated variance–covariance structures for random effects and errors can often lead to multiple solutions, which requires considering different sets of initial values.

By default, `menl` uses the EM algorithm to obtain initial values. This default routine works well in many cases but cannot be guaranteed to provide good initial values in all situations. Sometimes, you may need to specify your own initial values. Trying different initial values can also be useful to investigate the existence of multiple solutions and to verify convergence to a global maximum.

So far we have been "lucky" that all the examples worked without us having to specify initial estimates. You may not be that lucky with your data and model. So, in this section, we provide some guidance on how to find good initial values when the default initial values do not work well.

We present three approaches that you may choose to explore to find good initial estimates for the fixed effects. In some cases, you may also be able to obtain initial estimates for covariance parameters; see *Linearization approach to finding initial values*.

Linearization approach to finding initial values

Sometimes, we can use an LME model to obtain initial values of the NLME model by holding some of the parameters fixed at specific values. We can then fit the resulting LME model by using the `mixed` command and use the corresponding estimates as initial values for the NLME model. We refer to this initialization method as the linearization method.

We could have used this method in example 14 and example 19, if the default EM method did not provide reasonable initial estimates. In any case, it is good practice to specify different initial values to investigate potential convergence of the algorithm to a local maximum.

For instance, in example 14, we fit

$$\texttt{follicles}_{ij} = \phi_{1j} + \phi_{2j}\sin(2\pi\phi_{3j}\texttt{stime}_{ij}) + \phi_{4j}\cos(2\pi\phi_{3j}\texttt{stime}_{ij}) + \epsilon_{ij}$$

where

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\\ \phi_{4j}\end{bmatrix} = \begin{bmatrix}\beta_1 + u_{1j}\\ \beta_2\\ \beta_3\\ \beta_4\end{bmatrix}$$

This model is nonlinear because of the parameter ϕ_{3j}. To obtain initial values, we can hold ϕ_{3j} (or β_3) fixed at a specific value, say, $\beta_3 = 1$, thus making the above model linear,

$$\texttt{follicles}_{ij} = \phi_{1j} + \phi_{2j}\sin\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \phi_{4j}\cos\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

where

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\\ \phi_{4j}\end{bmatrix} = \begin{bmatrix}\beta_1 + u_{1j}\\ \beta_2\\ 1\\ \beta_4\end{bmatrix}$$

Or, more compactly,

$$\texttt{follicles}_{ij} = \beta_1 + u_{1j} + \beta_2\sin\left(2\pi\texttt{stime}_{ij}\right) + \beta_4\cos\left(2\pi\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

Now that the model is linear, we can use the `mixed` command to obtain initial values for β_1, β_2, and β_4 to be used in `menl`. In the code below, variables `sin1` and `cos1` are $\sin\left(2\pi\texttt{stime}_{ij}\right)$ and $\cos\left(2\pi\texttt{stime}_{ij}\right)$, respectively, and `|| mare:` specifies a random intercept at the `mare` level (see [ME] **mixed**). Also, for consistency with example 13, we assume an AR(1) within-group error correlation structure:

```
. mixed follicles sin1 cos1 || mare:, residuals(ar 1, t(time)) nolog
Mixed-effects ML regression                     Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
                                                Wald chi2(2)      =      39.00
Log likelihood = -776.51731                     Prob > chi2       =     0.0000
```

follicles	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
sin1	-2.958619	.4935054	-6.00	0.000	-3.925872	-1.991366
cos1	-.8798847	.5031764	-1.75	0.080	-1.866092	.1063228
_cons	12.18963	.9017435	13.52	0.000	10.42224	13.95701

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
mare: Identity				
var(_cons)	7.095503	3.764874	2.508047	20.07385
Residual: AR(1)				
rho	.5974664	.0547217	.4795551	.6941854
var(e)	13.08097	1.765326	10.04078	17.0417

```
LR test vs. linear model: chi2(2) = 242.63                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We will now use the estimates of the fixed effects shown in the output table as initial values for `menl` by specifying the `initial()` option. We use 1 as the initial value for `/phi3`. There are three ways to specify initial values in the `initial()` option; see *Specifying initial values*. Here we will use the specification where we repeatedly list a parameter name followed by its initial value; also see *Examples of specifying initial values*.

```
. local xb phi1:_cons 12.2 /phi2 -3.0 /phi3 1 /phi4 -.88
. menl follicles = {phi1: U1[mare], xb} + {phi2}*sin(2*_pi*stime*{phi3}) +
> {phi4}*cos(2*_pi*stime*{phi3}), rescorrelation(ar 1, t(time)) init(‘xb’)
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -775.627064
Iteration 2:    linearization log likelihood = -775.624331
Iteration 3:    linearization log likelihood = -775.624331
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
Linearization log likelihood = -775.62433
        phi1:  U1[mare], xb
```

follicles	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	12.18125	.9055136	13.45	0.000	10.40647	13.95602
/phi2	-2.874435	.5389255	-5.33	0.000	-3.930709	-1.81816
/phi3	.919119	.0512361	17.94	0.000	.8186981	1.01954
/phi4	-1.67526	.6766388	-2.48	0.013	-3.001447	-.3490722

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
mare: Identity				
var(U1)	7.207072	3.755605	2.595361	20.01336
Residual: AR(1), time time				
var(e)	12.63377	1.646898	9.785276	16.31146
corr	.5823733	.0544508	.4656902	.679153

In the above, we initialized only fixed-effects parameters and used naïve initial estimates of 1 for random-intercept and error variances and 0 for the correlation. We could have specified `initial()`'s `fixed` suboption to use the EM algorithm to compute initial estimates for the random-effects parameters; see *Examples of specifying initial values* for details.

With the linearization approach, we can also use estimates of the random-effects parameters from the `mixed` command to initialize the corresponding parameters of `menl`. This is an advantage of the linearization approach over the other two approaches we discuss in subsequent sections. One complication with the initialization of random-effects parameters is that the initial values must be supplied in the estimation metric, the metric used during estimation, instead of the parameter original metric. For example, instead of variances, we must supply estimates of log-standard deviations, and instead of covariances or correlations, we must supply inverse hyperbolic tangents of correlation parameters. Luckily for us, `mixed` uses the same estimation metric as `menl` and provides the `estmetric` option to display parameters in the estimation metric.

In our example, the random-effects parameters are the random-intercept variance, the within-group error variance, and the correlation between error terms. We refit the earlier `mixed` command but now with the `estmetric` option to obtain the estimates of the random-effects parameters in the estimation metric.

```
. mixed follicles sin1 cos1 || mare:, residuals(ar 1, t(time)) nolog estmetric
Mixed-effects ML regression                     Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
                                                Wald chi2(2)      =      39.00
Log likelihood = -776.51731                     Prob > chi2       =     0.0000
```

follicles	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
follicles						
sin1	-2.958619	.4935054	-6.00	0.000	-3.925872	-1.991366
cos1	-.8798847	.5031764	-1.75	0.080	-1.866092	.1063228
_cons	12.18963	.9017435	13.52	0.000	10.42224	13.95701
lns1_1_1						
_cons	.9797306	.2653			.5762499	1.665722
lnsig_e						
_cons	1.285579	.0674768	19.05	0.000	1.153327	1.417832
r_atr1						
_cons	.6891977	.0850992	8.10	0.000	.5224063	.8559892

`menl` uses the same ordering of the parameters as `mixed` does, so we can simply list all the estimates directly in the `initial()` option. When we list the values without parameter names, we must specify `initial()`'s `copy` suboption and specify the values for all parameters. In our example, we specify four fixed-effects coefficients and three random-effects parameters.

```
. menl follicles = {phi1: U1[mare], xb} + {phi2}*sin(2*_pi*stime*{phi3}) +
> {phi4}*cos(2*_pi*stime*{phi3}), rescorrelation(ar 1, t(time))
> initial(12.2 -3.0 1 -.88 .98 1.29 .69, copy)

Alternating PNLS/LME algorithm:

Iteration 1:    linearization log likelihood = -775.624332
Iteration 2:    linearization log likelihood = -775.624331

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
Linearization log likelihood = -775.62433

       phi1:  U1[mare], xb
```

follicles	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	12.18125	.9055135	13.45	0.000	10.40647	13.95602
/phi2	-2.874434	.5389241	-5.33	0.000	-3.930706	-1.818162
/phi3	.919119	.0512356	17.94	0.000	.818699	1.019539
/phi4	-1.675261	.6766409	-2.48	0.013	-3.001452	-.3490689

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
mare: Identity				
var(U1)	7.207072	3.755605	2.595361	20.01336
Residual: AR(1), time time				
var(e)	12.63377	1.646898	9.785276	16.31146
corr	.5823733	.0544508	.4656903	.679153

The results are different from those in example 14. The value of the linearization log likelihood in this example, −775.62, is larger than that from example 14, −789.43. So it appears that we have converged to a local maximum of the linearization log likelihood in example 14.

Our initial values based on `mixed` turned out to be better than those computed by default by `menl`. This is not surprising. In general, `menl`'s EM algorithm should produce reasonable initial values for many nonlinear models, but the initial values may not necessarily be optimal for all of those models. In this example, our initial values were tailored to the ovary data and the model.

In general, sensitivity to initial values is one of the key issues in NLME models, especially for models that involve periodic functions. Therefore, it is important to try different sets of initial values to verify global convergence before reporting your final results. Sometimes, you may even have to rely on your knowledge of the science behind the problem to decide which set of results is more reasonable.

Graphical approach to finding initial values

If your model has parameters that have natural physical interpretations, you may be able to obtain starting values from a graph of the data.

Draper and Smith (1998) presented a dataset in which the trunk circumference `circumf` (in mm) of five different orange trees was measured over seven different time points, stored in `age`. Pinheiro and Bates (2000) suggested the following model for these data:

$$\texttt{circumf}_{ij} = \frac{\phi_{1j}}{1 + \exp\left\{-\left(\texttt{age}_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij} \tag{21}$$

In this model, ϕ_{1j} is the asymptotic trunk circumference for the jth tree as $\texttt{age}_{ij} \to \infty$, ϕ_{2j} is the age at which the jth tree attains half of its asymptotic trunk circumference ϕ_{1j}, and ϕ_{3j} is a scale parameter; see the graph below.

The stage 2 specification of this model is

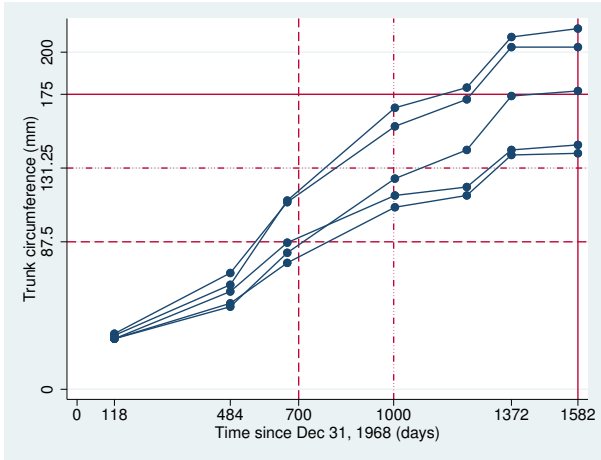
$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 + u_{1j} \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

where

$$u_{1j} \sim N\left(0, \sigma^2_{u_1}\right),\ \epsilon_{ij} \sim N\left(0, \sigma^2_{\epsilon}\right)$$

Because the model parameters have graphical interpretations, we can plot our data and obtain initial values from the graph.

```
. use http://www.stata-press.com/data/r15/orange
(Growth of orange trees (Draper and Smith, 1998))
. twoway connected circumf age, connect(L) yline(175) xline(1582)
> yline(87.5, lpattern(dash)) xline(700, lpattern(dash))
> yline(131.25, lpattern("-...")) xline(1000, lpattern("-..."))
> xlabel(0 118 484 700 1000 1372 1582) ylabel(#5 87.5 131.25 175)
```



From the above graph, the mean asymptotic trunk circumference can be estimated as 175 mm, which is roughly the mean of the circumference values at age 1,582 (in days). The trees attain half of their asymptotic trunk circumference, $175/2 = 87.5$, at about age 700 (in days). Therefore, we use the initial estimates $\beta_1 = 175$ for the asymptotic trunk circumference and $\beta_2 = 700$ for the location of the inflection point. To obtain an initial estimate for β_3, we note that when $\texttt{age} = \beta_2 + \beta_3$ in (21), $E(\texttt{circumf}_{ij}) = \beta_1/\{1 + \exp(-1)\} = 0.73\beta_1$, which we will approximate as $0.75\beta_1$ for the purpose of the graph. That is, the logistic curve reaches approximately 3/4 of its asymptotic value, $0.75 \times 175 = 131.25$, at $\texttt{age} = \beta_2 + \beta_3$. The above graph suggests that the trees attain 3/4 of their final trunk circumference at about 1,000 days ($= \beta_2 + \beta_3$), giving an initial estimate of $\beta_3 = 1000 - 700 = 300$. We can now supply these values to `menl` in the `initial()` option.

Unfortunately, the graph does not provide us with the estimates for variance components. In this case, we can use `initial()`'s `fixed` suboption to specify that the EM algorithm still be used to initialize variance components, while the supplied values be used to initialize fixed effects. If we do not specify `fixed`, `menl` will use naïve initial estimates for variance components such as ones for variances and zeros for covariances.

We now fit the model using our own initial estimates for fixed effects:

```
. menl circumf = {phi1: U1[tree], xb}/(1+exp(-(age-{phi2})/{phi3})),
> initial(phi1:_cons 175 /phi2 700 /phi3 300, fixed)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -131.584945
Iteration 2:    linearization log likelihood = -131.584579
Iteration 3:    linearization log likelihood = -131.584579
Computing standard errors:
```

```
Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
Linearization log likelihood = -131.58458
       phi1:  U1[tree], xb
```

circumf	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	191.049	16.15403	11.83	0.000	159.3877	222.7103
/phi2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/phi3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
tree: Identity				
var(U1)	991.1514	639.4636	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

For comparison, we fit the same model but now using the default initial values for fixed effects:

```
. menl circumf = {phi1: U1[tree], xb}/(1+exp(-(age-{phi2})/{phi3}))
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:    linearization log likelihood = -131.584579
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
Linearization log likelihood = -131.58458
       phi1:  U1[tree], xb
```

circumf	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
phi1						
_cons	191.049	16.15403	11.83	0.000	159.3877	222.7103
/phi2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/phi3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
tree: Identity				
var(U1)	991.1514	639.4636	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

The results are identical except for the iteration log.

Smart regressions approach to finding initial values

Consider the following NLME model,

$$\mathtt{y}_{ij} = \phi_{1j} + (\phi_{2j} - \phi_{1j})\, \exp\{-\exp(\phi_{3j})\, \mathtt{x}_{ij}\} + \epsilon_{ij}$$

where

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 + u_{1j} \\ \beta_3 \end{bmatrix}$$

Here ϕ_{1j} is the asymptote as $\mathtt{x}_{ij} \to \infty$ and ϕ_{2j} is the value of $\mathtt{y}_{ij}$ at $\mathtt{x}_{ij} = 0$. Thus initial estimates, $\beta_1^{(0)}$ and $\beta_2^{(0)}$, may be obtained by using the graphical approach as described in *Graphical approach to finding initial values*. To obtain an initial estimate for β_3, notice that, ignoring the error term ϵ_{ij} and setting $u_{1j} = 0$,

$$\log(|\mathtt{y}_{ij} - \beta_1|) = \log(\beta_2 - \beta_1) + \{-\exp(\beta_3)\}\, \mathtt{x}_{ij}$$

Therefore, we can regress $\log(|\mathtt{y} - \beta_1^{(0)}|)$ on `x` and use the estimated slope, $\widehat{\beta}_x = -\exp(\beta_3^{(0)})$, to obtain the initial value for $\beta_3^{(0)} = \log(-\widehat{\beta}_x)$.

Examples of specifying initial values

When you want to assign initial values for a subset of the model parameters, for example, fixed effects or random-effects covariance parameters, you will often need to know their estimation names or, in other words, how `menl` labels them in `e(b)`. To learn the names, you can fit the model with the `iterate(0)` and `coeflegend` options first.

```
. menl ... , ... iterate(0) coeflegend
```

The `iterate(0)` option specifies to bypass maximization and only report the initial values and the likelihood evaluated at those values. The `coeflegend` option specifies that the legend of the parameters and how to specify them in an expression be displayed rather than displaying the statistics for the parameters.

Keep in mind, however, that `menl` does not perform estimation in the original parameter metric. For computational stability, the estimation is performed, loosely speaking, in a metric that transforms all parameters to be defined on a real line. For example, a log transformation is used for standard deviations, and an inverse hyperbolic tangent transformation is used for correlations. When you specify initial values, you must specify them for parameters in the estimation metric and not the original metric.

`coeflegend` displays parameter names as they are stored in `e(b)`, which, for `menl`, are the names of estimation parameters. If you also want to see parameters in the original metric, you can specify `coeflegend` on replay.

```
. menl ... , ... iterate(0)
. menl, coeflegend
```

For example, recall the NLME model for the soybean data from example 9. Suppose that we want to supply our own initial values.

We fit the model with `iterate(0)` and `coeflegend`:

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: U1[plot], xb)
> define(phi2: U2[plot], xb)
> define(phi3: U3[plot], xb)
> covariance(U*, unstructured) iterate(0) coeflegend
Obtaining starting values by EM:

Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10
Linearization log likelihood = -740.06177
        phi1:  U1[plot], xb
        phi2:  U2[plot], xb
        phi3:  U3[plot], xb
```

weight	Coef.	Legend
phi1		
_cons	19.26527	_b[phi1:_cons]
phi2		
_cons	55.05299	_b[phi2:_cons]
phi3		
_cons	8.385531	_b[phi3:_cons]
/plot		
lnsd(U1)	1.650846	_b[/plot:lnsd(U1)]
lnsd(U2)	1.436634	_b[/plot:lnsd(U2)]
lnsd(U3)	.4081525	_b[/plot:lnsd(U3)]
athcorr(U2, U1)	.9055785	_b[/plot:athcorr(U2,U1)]
athcorr(U3, U1)	.8482105	_b[/plot:athcorr(U3,U1)]
athcorr(U3, U2)	1.537798	_b[/plot:athcorr(U3,U2)]
/Residual		
lnsigma	.1069986	_b[/Residual:lnsigma]

```
Warning: convergence not achieved
```

Parameter names are listed within the `_b[]` specifier.

In what follows, we will outline only the syntax of the specifications. If you actually want to run all the examples to see the initialization in action, we suggest that you specify `iterate(0)` for speed.

Let's first specify initial values for fixed effects only. The fixed-effects parameters are `phi1:_cons`, `phi2:_cons`, and `phi3:_cons`. Suppose that we want to initialize them with 19, 55, and 8.

We can type

```
. menl ..., ... initial(phi1:_cons 19 phi2:_cons 55 phi3:_cons 8)
```

Or, more compactly, we can type

```
. local fe phi1:_cons 19 phi2:_cons 55 phi3:_cons 8
. menl ..., ... initial(‘fe’)
```

When you specify the `initial()` option, `menl` does not perform the EM algorithm to initialize the parameters but instead uses the values you supplied. If you specify values for only a subset of parameters, the remaining parameters will be initialized with naïve initial values such as zeros for fixed effects and correlations and ones for variances. Often, you may have good initial values for fixed effects but not for variance components. In this situation, `menl` provides `initial()`'s `fixed` suboption. This option specifies that the supplied values be used for fixed effects but that the EM algorithm still be used to obtain initial values for variance components. If you specify only a subset of values for fixed effects, the remaining fixed effects will still be initialized with zeros even if `fixed` is specified. We recommend that you specify `fixed` when you intend to supply initial values only for the fixed effects.

```
. local fe phi1:_cons 19 phi2:_cons 55 phi3:_cons 8
. menl ..., ... initial('fe', fixed)
```

Now suppose that we also want to assign initial values for random-effects parameters. As we mentioned earlier, remember that we assign initial values for standard deviations in the log metric and for correlation in the inverse hyperbolic tangent or atanh metric. For example, if you want to assign an initial value of 2 to σ_ϵ, then you should supply $\log(2)$ to the `initial()` option. Similarly, if you want to assign a value of 0.7 to the correlation of two random effects, then you should provide `atanh`(0.7) to the `initial()` option.

Continuing with example 9, suppose that we want to specify the following initial values for the random-effects covariance parameters:

$$\begin{array}{c} \texttt{U1[plot]} \quad \texttt{U2[plot]} \quad \texttt{U3[plot]} \\ \begin{pmatrix} \sigma_1 = 5 & & \\ \rho_{21} = 0.72 & \sigma_2 = 4 & \\ \rho_{31} = 0.71 & \rho_{32} = 0.94 & \sigma_3 = 1.4 \end{pmatrix} \end{array}$$

The names of the parameters in the estimation metric that correspond to σ_1, σ_2, and σ_3 are `/plot:lnsd(U1)`, `/plot:lnsd(U2)`, and `/plot:lnsd(U3)` and that correspond to ρ_{21}, ρ_{31}, and ρ_{32} are `/plot:athcorr(U2,U1)`, `/plot:athcorr(U3,U1)`, and `/plot:athcorr(U3,U2)`.

When specifying initial values for free parameters such as random-effects covariance parameters, you can omit the forward slash (`/`) at the beginning of their names. Keeping in mind that initial values for covariance parameters are supplied in the log and atanh metrics, we can type

```
. local re_cov          plot:lnsd(U1) log(5)            // log(5)
. local re_cov 're_cov' plot:lnsd(U2) 1.4               // log(4)
. local re_cov 're_cov' plot:lnsd(U3) 0.34              // log(1.4)
. local re_cov 're_cov' plot:athcorr(U2,U1) atanh(0.72) // atanh(0.72)
. local re_cov 're_cov' plot:athcorr(U3,U1) 0.89        // atanh(0.71)
. local re_cov 're_cov' plot:athcorr(U3,U2) 1.7         // atanh(0.94)

. menl ... , ... initial('fe' 're_cov' Residual:lnsigma 0.5)
```

In the above, we also specified an initial value of 0.5 for the log of the error standard deviation. For parameters `/plot:lnsd(U1)` and `/plot:athcorr(U2,U1)`, instead of specifying the values, we specified the corresponding expression. This is allowed, as long as your expression is simple and does not contain spaces.

Instead of using parameter names, we can specify a list of values directly in the `initial()` option, in which case we must also specify `initial()`'s `copy` suboption.

```
. menl ... , ... initial(19 55 8 1.6 1.4 0.34 0.9 0.89 1.7 0.5, copy)
```

Or we can provide these values as a matrix:

```
. matrix initvals = (19, 55, 8, 1.6, 1.4, 0.34, 0.9, 0.89, 1.7, 0.5)
. matrix list initvals
initvals[1,10]
     c1   c2   c3   c4   c5   c6   c7   c8   c9  c10
r1   19   55    8  1.6  1.4  .34   .9  .89  1.7   .5
. menl ... , ... initial(initvals, copy)
```

If we label the columns of the `initvals` matrix properly, we do not need to specify `copy`:

```
. local fullcolnames : colfullnames e(b)
. matrix colnames initvals = ‘fullcolnames’
. matrix list initvals
initvals[1,10]
          phi1:        phi2:        phi3:       /plot:       /plot:
          _cons        _cons        _cons     lnsd(U1)     lnsd(U2)
r1           19           55            8          1.6          1.4
          /plot:       /plot:       /plot:       /plot:   /Residual:
                athcorr(U2,  athcorr(U3,  athcorr(U3,
       lnsd(U3)          U1)          U1)          U2)      lnsigma
r1          .34           .9          .89          1.7           .5
. menl ... , ... initial(initvals)
```

Using a properly labeled initial-value matrix, we can also specify initial values for a subset of parameters. For example, we can specify initial values for fixed effects only as follows:

```
. matrix initvals = initvals[1,1..3]
. matrix list initvals
initvals[1,3]
     phi1:  phi2:  phi3:
    _cons  _cons  _cons
r1     19     55      8
. menl ... , ... initial(initvals)
```

Stored results

`menl` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_f)`	number of fixed-effects parameters
`e(k_r)`	number of random-effects parameters
`e(k_rs)`	number of variances
`e(k_rc)`	number of covariances
`e(k_res)`	number of within-group error parameters
`e(k_eq)`	number of equations
`e(k_feq)`	number of fixed-effects equations
`e(k_req)`	number of random-effects equations
`e(k_reseq)`	number of within-group error equations
`e(df_m)`	model degrees of freedom
`e(ll)`	linearization log (restricted) likelihood
`e(chi2)`	χ^2
`e(p)`	significance
`e(rank)`	rank of `e(V)`
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

e(cmd)	menl
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(ivars)	grouping variables
e(title)	title in estimation output
e(varlist)	variables used in the specified equation
e(eq_*depvar*)	user-specified equation
e(expressions)	names of defined expressions, *expr_1*, *expr_2*,..., *expr_k*
e(ex_*expr_i*)	defined expression *expr_i*, i=1,...,k
e(hierarchy)	random-effects hierarchy structure, (*path*:*covtype*:*REs*) (...)
e(revars)	names of random effects
e(rstructlab)	within-group error covariance output label
e(timevar)	within-group error covariance t() variable, if specified
e(indexvar)	within-group error covariance index() variable, if specified
e(covbyvar)	within-group error covariance by() variable, if specified
e(stratavar)	within-group error variance strata() variable, if specified
e(corrbyvar)	within-group error correlation by() variable, if specified
e(rescovopt)	within-group error covariance option, if rescovariance() specified
e(resvaropt)	within-group error variance option, if resvariance() specified
e(rescorropt)	within-group error correlation option, if rescorrelation() specified
e(chi2type)	Wald; type of model χ^2 test
e(vce)	conventional
e(method)	ML or REML
e(opt)	type of optimization, lbates
e(crittype)	optimization criterion, linearization log likelihood
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(marginsdefault)	default predict() specification for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	factor-variable constraint matrix
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(b_sd)	random-effects and within-group error estimates in the standard deviation metric
e(V_sd)	VCE for parameters in the standard deviation metric
e(b_var)	random-effects and within-group error estimates in the variance metric
e(V_var)	VCE for parameters in the variance metric
e(cov_#)	random-effects covariance structure at the hierarchical level $k-\#+1$ in a k-level model
e(hierstats)	group-size statistics for each hierarchy

Functions

e(sample)	marks estimation sample

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction
Variance-components parameters
Inference based on linearization
Initial values

Introduction

Recall (1), a two-level NLME model, from the *Introduction*,

$$y_{ij} = \mu\left(\mathbf{x}'_{ij}, \boldsymbol{\beta}, \mathbf{u}_j\right) + \epsilon_{ij} \quad i = 1, \ldots, n_j;\ j = 1, \ldots, M$$

where M is the number of clusters and, for each cluster j, n_j is the number of observations in that cluster; $\mathbf{y}_j = (y_{1j}, y_{2j}, \ldots, y_{n_j j})'$ is the $n_j \times 1$ response vector; $\mathbf{X}_j = (\mathbf{x}_{1j}, \mathbf{x}_{2j}, \ldots, \mathbf{x}_{n_j j})'$ is the $n_j \times l$ matrix of covariates, including within-subject and between-subjects covariates; $\boldsymbol{\beta}$ is the $p \times 1$ vector of unknown parameters; $\mathbf{u}_j$ is the $q \times 1$ vector of random effects; and $\boldsymbol{\epsilon}_j = (\epsilon_{1j}, \epsilon_{2j}, \ldots, \epsilon_{n_j j})'$ is the $n_j \times 1$ vector of within-group or within-cluster errors. $\mathbf{u}_j$'s follow a multivariate normal distribution with mean 0 and $q \times q$ variance–covariance matrix $\boldsymbol{\Sigma}$, and $\boldsymbol{\epsilon}_j$'s follow a multivariate normal distribution with mean 0 and $n_j \times n_j$ variance–covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$; $\mathbf{u}_j$'s are assumed to be independent of $\boldsymbol{\epsilon}_j$'s. Depending on the form of $\boldsymbol{\Lambda}_j$, σ^2 is either a within-group error variance σ^2_ϵ or a squared scale parameter σ^2. For example, when errors are i.i.d., that is, when $\boldsymbol{\Lambda}_j$ is the identity matrix, $\sigma^2 = \sigma^2_\epsilon$ is the within-group error variance. When $\boldsymbol{\Lambda}_j$ corresponds to the heteroskedastic power structure, σ^2 is a multiplier or a scale parameter.

Positive-definite matrices $\boldsymbol{\Sigma}/\sigma^2$ and $\boldsymbol{\Lambda}_j$ are expressed as functions of unconstrained parameter vectors $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_w$, respectively, to recast a constrained optimization problem into an unconstrained one. Thus $\boldsymbol{\alpha}_u$ contains unconstrained random-effects covariance parameters and $\boldsymbol{\alpha}_w$ contains unconstrained within-group error covariance parameters. $\boldsymbol{\Lambda}_j$ may also depend on the random effects $\mathbf{u}_j$ and the fixed effects $\boldsymbol{\beta}$. For more details about $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}_j$ and about functional forms of parameter vectors $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_w$ given different covariance structures, see *Variance-components parameters*.

Based on (1), the marginal, with respect to $\mathbf{u}_j$'s, log likelihood for $\left(\boldsymbol{\beta}, \boldsymbol{\alpha}, \sigma^2\right)$ is

$$L(\boldsymbol{\beta}, \boldsymbol{\alpha}, \sigma^2) = \log\left\{\prod_{j=1}^{M} \int f\left(\mathbf{y}_j | \mathbf{X}_j, \mathbf{u}_j; \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \sigma^2\right) f\left(\mathbf{u}_j; \boldsymbol{\alpha}_u\right) d\mathbf{u}_j\right\} \tag{22}$$

where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}'_u, \boldsymbol{\alpha}'_w)'$, $f\left(\mathbf{y}_j | \mathbf{X}_j, \mathbf{u}_j; \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \sigma^2\right)$ is the conditional density of $\mathbf{y}_j$ given $\mathbf{X}_j$ and $\mathbf{u}_j$, and $f\left(\mathbf{u}_j; \boldsymbol{\alpha}_u\right)$ is the density of $\mathbf{u}_j$.

In general, there are no closed-form expressions for (22) or the marginal moments of an NLME model. This is because the random effects $\mathbf{u}_j$ enter the model nonlinearly, making the q-dimensional integral in (22) analytically intractable in all but simpler cases. Several estimation techniques have been proposed for estimating parameters $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$, and σ^2, including numerical integration of the integral in (22) by using an adaptive Gaussian quadrature and a linearization of the mean function in (1) by using a Taylor-series expansion.

`menl` implements the linearization method of Lindstrom and Bates (1990), with extensions from Pinheiro and Bates (1995), which is described in *Inference based on linearization*.

Variance-components parameters

For numerical stability, maximization of (22) is performed with respect to the unique elements of the matrix $\boldsymbol{G} = \boldsymbol{\Sigma}/\sigma^2$ expressed as logarithms of standard deviations for the diagonal elements and hyperbolic arctangents of the correlations for off-diagonal elements. Let $\boldsymbol{\alpha}_u$ be the vector containing these elements. For example, if we assume that the elements of the random-effects vector $\mathbf{u}_j$ are independent, then $\boldsymbol{\Sigma}$ is diagonal and $\boldsymbol{\alpha}_u$ will contain q distinct parameters—q logarithms of standard deviations. Table 1 lists the vectors of parameters $\boldsymbol{\alpha}_u$ for all random-effects covariance structures supported by `menl` in the `covariance(`*vartype*`)` option.

Table 1. Variance-components parameters

vartype	$\boldsymbol{\alpha}'_u$
`independent`	$(g_1, g_2, \ldots, g_q)$
`exchangeable`	(g_1, g_{12})
`identity`	g_1
`unstructured`	$(g_1, g_2, \ldots, g_q, g_{12}, g_{13}, \ldots, g_{q-1q})$

Notes: g_u=log($\sqrt{[\mathbf{G}]_{uu}}$), g_{uv}= `atanh`($[\mathbf{G}]_{uv}$).
`unstructured` has $q(q+1)/2$ parameters.

The within-group error covariance matrix is parameterized as follows,

$$\text{Var}\left(\boldsymbol{\epsilon}_j|\mathbf{u}_j\right) = \sigma^2\boldsymbol{\Lambda}_j\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j, \boldsymbol{\alpha}_w\right) = \sigma^2\mathbf{S}_j\left(\boldsymbol{\delta}, \boldsymbol{\upsilon}_j\right)\mathbf{C}_j(\boldsymbol{\rho})\mathbf{S}_j\left(\boldsymbol{\delta}, \boldsymbol{\upsilon}_j\right)$$

where $\boldsymbol{\alpha}_w = \left(\boldsymbol{\delta}^{*\prime}, \boldsymbol{\rho}^{*\prime}\right)'$ and $\boldsymbol{\delta}^*$ and $\boldsymbol{\rho}^*$ are unconstrained versions of $\boldsymbol{\delta}$ and $\boldsymbol{\rho}$ defined in table 2 and table 3, respectively. For example, for a positive δ_1, $\delta_1^* = \log\left(\delta_1\right)$. $\mathbf{S}_j = \mathbf{S}_j\left(\boldsymbol{\delta}, \boldsymbol{\upsilon}_j\right)$ is an $n_j \times n_j$ diagonal matrix with nonnegative diagonal elements $g\left(\boldsymbol{\delta}, \upsilon_{1j}\right), g\left(\boldsymbol{\delta}, \upsilon_{2j}\right), \ldots, g\left(\boldsymbol{\delta}, \upsilon_{n_j j}\right)$ such that $\text{Var}\left(\epsilon_{ij}\right) = \sigma^2[\mathbf{S}_j]^2_{ii} = \sigma^2 g^2\left(\boldsymbol{\delta}, \upsilon_{ij}\right)$, where υ_{ij}'s are the values of a variance covariate or the values of a mean function $\mu\left(\mathbf{x}'_{ij}, \boldsymbol{\beta}, \mathbf{u}_j\right)$, in which case $\boldsymbol{\Lambda}_j$ will depend on $\mathbf{X}_j$, $\boldsymbol{\beta}$, and $\mathbf{u}_j$. $\mathbf{C}_j = \mathbf{C}_j(\boldsymbol{\rho})$ is a correlation matrix such that $\text{corr}\left(\epsilon_{ij}, \epsilon_{kj}\right) = [\mathbf{C}_j]_{ik} = h\left(|t_{ij} - t_{kj}|, \boldsymbol{\rho}\right)$, where t_{ij} is a value of a time variable for time-dependent correlation structures such as AR and MA structures. A list of the supported $g(\cdot)$ and $h(\cdot)$ functions is given in table 2 and table 3, respectively.

Carroll and Ruppert (1988) introduced various variance functions $g\left(\boldsymbol{\delta}, \upsilon_{ij}\right)$ to model heteroskedasticity, which were further studied in the context of NLME models by Davidian and Giltinan (1995). Table 2 lists variance functions supported by the `resvariance(`*resvarfunc* `...)` option.

Table 2. Supported variance functions $g(\cdot)$

resvarfunc	$g\left(\boldsymbol{\delta}, \upsilon_{ij}\right)$	$\boldsymbol{\delta}'$
`identity`	1	–
`linear`	$\sqrt{\upsilon_{ij}}$	–
`power`	$c + \lvert\upsilon_{ij}\rvert^{\delta}$	(c, δ), $c \geq 0$
`power, noconstant`	$\lvert\upsilon_{ij}\rvert^{\delta}$	δ
`exponential`	$\exp\left(\delta\upsilon_{ij}\right)$	δ

The variance function $g(\cdot)$ and thus the within-group error covariance may depend on $\boldsymbol{\beta}$ and $\mathbf{u}_j$ through $\mu(\cdot)$, when $\upsilon_{ij} = \mu_{ij} = \mu\left(\mathbf{x}'_{ij}, \mathbf{u}_j, \boldsymbol{\beta}\right)$ in table 2. This is particularly appealing in PK applications, where there is often considerable intraindividual heterogeneity that is modeled, for example, as a power function of the mean.

The within-group error correlation structure is governed by the $h(\cdot)$ function. Table 3 lists correlation structures that are supported by the `rescorrelation(`*rescorr* `...)` option and also have a closed-form expression. In addition, the AR and MA correlation structures are defined below.

The `ar` p structure assumes that the errors have an AR structure of order p. That is,

$$\epsilon_{ij} = \phi_1\epsilon_{i-1,j} + \cdots + \phi_p\epsilon_{i-p,j} + z_{ij}$$

where z_{ij} are i.i.d. Gaussian with mean 0 and variance σ_z^2. `menl` reports estimates of $\phi_1, \ldots, \phi_p$ and the overall error variance σ_ϵ^2, which can be derived from the above expression. This structure has a closed-form expression only for $p = 1$, in which case $\phi_1 = \rho$ is the correlation between error terms.

The `ma` q structure assumes that the errors are an MA process of order q. That is,

$$\epsilon_{ij} = Z_i + \theta_1 Z_{i-1} + \cdots + \theta_q Z_{i-q}$$

where Z_l are i.i.d. Gaussian with mean 0 and variance σ_Z^2. `menl` reports estimates of $\theta_1, \ldots, \theta_q$ and the overall error variance σ_ϵ^2, which can be derived from the above expression.

Table 3. Within-group error correlation functions $h(\cdot)$

rescorr	$h(\lvert t_{ij} - t_{kj}\rvert, \boldsymbol{\rho})$	Expression	$\boldsymbol{\rho}$
`identity`	$h(k)$	$I(k = 0)$	–
`ar 1`	$h(k, \rho)$	$\rho^k,\ k = 0, 1, \ldots$	$\rho,\ \lvert\rho\rvert < 1$
`ar` p, $p > 1$	$h(k, \boldsymbol{\phi})$	no closed form	$(\phi_1, \phi_2, \ldots, \phi_p)$
`ctar1`	$h(s, \rho)$	$\rho^s,\ s \geq 0$	$\rho,\ \lvert\rho\rvert < 1$
`ma` q	$h(k, \boldsymbol{\theta})$	$\begin{cases} \frac{\sum_{j=0}^{q-\lvert k\rvert} \theta_j\theta_{j+\lvert k\rvert}}{\sum_{j=0}^{q}\theta_j^2} & k \leq q \\ 0 & k > q \end{cases}$	$(\theta_0 = 1, \theta_1, \ldots, \theta_q)$

You can build many flexible within-group error covariance structures by combining different functions $g(\cdot)$ and $h(\cdot)$, that is, by combining the `resvariance()` and `rescorrelation()` options. For example, you can combine an AR(1) correlation structure with a heteroskedastic structure that is expressed as a power function of the mean by specifying `rescorrelation(ar 1, t(timevar))` and `resvariance(power _yhat)`.

Inference based on linearization

Let's write (1), equivalently, in matrix form as

$$\mathbf{y}_j = \boldsymbol{\mu}\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j\right) + \boldsymbol{\Lambda}_j^{\frac{1}{2}}\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j, \boldsymbol{\alpha}_w\right)\boldsymbol{e}_j$$

Here $\boldsymbol{\mu}\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j\right)$ depends on $\boldsymbol{\beta}$ and $\mathbf{u}_j$ through the function $\boldsymbol{d}(\cdot)$ in (2), and $\boldsymbol{e}_j$'s $\sim N\left(\mathbf{0}, \sigma^2 I_{n_j}\right)$, where I_{n_j} is the identity matrix of dimension n_j. In what follows, for brevity, we suppress the dependence of $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}_j$ on $\mathbf{X}_j$.

Following Lindstrom and Bates (1990), we will initially assume that $\boldsymbol{\Lambda}_j$ does not depend on $\mathbf{X}_j$, $\boldsymbol{\beta}$, and $\mathbf{u}_j$ or, equivalently, on $\boldsymbol{\phi}_j$ but rather on j only through its dimension; that is, $\boldsymbol{\Lambda}_j = \boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)$. Therefore, heteroskedastic structures that depend on the mean are not yet allowed in this context. Toward the end of this section, we will present a modified version of the algorithm that accounts for the dependence of $\boldsymbol{\Lambda}_j$ on $\boldsymbol{\phi}_j$.

Lindstrom and Bates discuss a natural extension of the methods for the LME models to NLME models. For a known $\boldsymbol{\alpha}$ (and thus known $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}_j$) and σ^2, the estimates of $\boldsymbol{\beta}$ and $\mathbf{u}_j$ jointly minimize

$$\sum_{j=1}^{M}\Big[\log|\boldsymbol{\Sigma}(\boldsymbol{\alpha}_u)| + \mathbf{u}_j'\{\boldsymbol{\Sigma}(\boldsymbol{\alpha}_u)\}^{-1}\mathbf{u}_j + \log\left|\sigma^2\boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)\right| + \sigma^{-2}\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}'\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{\alpha}_w)\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}\Big]$$

which is twice the negative log likelihood for $\boldsymbol{\beta}$ when $\mathbf{u}_j$ is fixed or twice the negative log of the posterior density of $\mathbf{u}_j$ when $\boldsymbol{\beta}$ is fixed. Consequently, one strategy for estimating $\boldsymbol{\beta}$ and (predicting) $\mathbf{u}_j$ is to minimize the above objective function with respect to $\boldsymbol{\beta}$ and $\mathbf{u}_j$ given suitable estimates of $\boldsymbol{\alpha}$ and σ^2. Estimation of $\boldsymbol{\alpha}$ and σ^2 can be accomplished by using MLE with respect to the marginal density of $\mathbf{y}_j$, in which $\mathbf{u}_j$'s are integrated out. But because no closed-form expression for this density is available, we approximate the conditional distribution of $\mathbf{y}_j$ given $\mathbf{u}_j$ by a multivariate normal distribution with an expectation that is linear in $\mathbf{u}_j$ and $\boldsymbol{\beta}$. This is illustrated in step 2 of the algorithm below.

Lindstrom and Bates (1990) propose the following two-step estimation method or alternating algorithm.

Step 1 (PNLS step). Given current estimates $\widehat{\boldsymbol{\alpha}}$ (and thus $\widehat{\boldsymbol{\alpha}}_u$ and $\widehat{\boldsymbol{\alpha}}_w$) of $\boldsymbol{\alpha}$ and $\widehat{\sigma}^2$ of σ^2, minimize with respect to $\boldsymbol{\beta}$ and $\mathbf{u}_j$

$$\sum_{j=1}^{M}\Big[\log|\boldsymbol{\Sigma}(\widehat{\boldsymbol{\alpha}}_u)| + \mathbf{u}_j'\{\boldsymbol{\Sigma}(\widehat{\boldsymbol{\alpha}}_u)\}^{-1}\mathbf{u}_j + \log\left|\widehat{\sigma}^2\boldsymbol{\Lambda}_j(\widehat{\boldsymbol{\alpha}}_w)\right| + \widehat{\sigma}^{-2}\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}'\boldsymbol{\Lambda}_j^{-1}(\widehat{\boldsymbol{\alpha}}_w)\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}\Big] \tag{23}$$

Define $\boldsymbol{\Delta}$ such that $\sigma^2\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Delta}'\boldsymbol{\Delta}$. Note that $\boldsymbol{\Delta} = \boldsymbol{\Delta}(\boldsymbol{\alpha}_u)$, but for notational convenience, this dependency is suppressed throughout the rest of this section. Equation (23) is equivalent to minimizing the penalized least-square objective function

PNLS step: $$\sum_{j=1}^{M}\left[\left|\left|\left\{\boldsymbol{\Lambda}_j'(\boldsymbol{\alpha}_w)\right\}^{-1/2}\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}\right|\right|^2 + ||\boldsymbol{\Delta}\mathbf{u}_j||^2\right]$$

with respect to $\boldsymbol{\beta}$ and $\mathbf{u}_j$ while holding the current estimates of $\boldsymbol{\alpha}$ (and, consequently, of $\boldsymbol{\Delta}$ and of $\boldsymbol{\Lambda}_j$) fixed. `pnlsopts(iterate(#))` iterations are performed at this step, unless the convergence criterion (CC) is met. The CC for PNLS optimization is controlled by `pnlsopts(nrtolerance(#))` and one of `pnlsopts(ltolerance(#))` or `pnlsopts(tolerance(#))`; see *menlmaxopts* for details.

Denote the resulting estimates as $\widehat{\mathbf{u}}_j$ and $\widehat{\boldsymbol{\beta}}$.

Step 2 (LME step). Perform a first-order Taylor-series expansion of the model mean function around the current estimates of $\boldsymbol{\beta}$ and of the conditional modes of the random effects $\mathbf{u}_j$, yielding

$$\mathbf{y}_j = \boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}},\widehat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + \widehat{\mathbf{Z}}_j(\mathbf{u}_j - \widehat{\mathbf{u}}_j) + \boldsymbol{\Lambda}_j^{\frac{1}{2}}(\boldsymbol{\alpha}_w)\,\boldsymbol{e}_j \tag{24}$$

where

$$\widehat{\mathbf{X}}_j = \left.\frac{\partial \boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)}{\partial \boldsymbol{\beta}'}\right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}, \mathbf{u}_j=\widehat{\mathbf{u}}_j}$$
$$\widehat{\mathbf{Z}}_j = \left.\frac{\partial \boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)}{\partial \mathbf{u}_j'}\right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}, \mathbf{u}_j=\widehat{\mathbf{u}}_j}$$

Model (24) is essentially an LME model, and we use notations $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Z}}_j$ for the derivatives to emphasize this. That is, $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Z}}_j$ represent the corresponding fixed-effects and random-effects design matrices of an LME model.

Thus the approximate conditional distribution of $\mathbf{y}_j$ is

$$\mathbf{y}_j|\mathbf{u}_j \sim N\left\{\boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + \widehat{\mathbf{Z}}_j(\mathbf{u}_j - \widehat{\mathbf{u}}_j),\, \sigma^2\boldsymbol{\Lambda}_j\right\}$$

Because the expectation is now linear in random effects $\mathbf{u}_j$, the approximate conditional distribution of $\mathbf{y}_j$, along with distribution of $\mathbf{u}_j$, allows us to approximate the marginal distribution of $\mathbf{y}_j$ as

$$\mathbf{y}_j \sim N\left\{\boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) - \widehat{\mathbf{Z}}_j\widehat{\mathbf{u}}_j,\, \sigma^2\mathbf{V}_j(\boldsymbol{\alpha})\right\} \tag{25}$$

where $\mathbf{V}_j(\boldsymbol{\alpha}) = \widehat{\mathbf{Z}}_j\boldsymbol{\Delta}^{-1}\left(\boldsymbol{\Delta}^{-1}\right)'\widehat{\mathbf{Z}}_j{}' + \boldsymbol{\Lambda}_j\left(\boldsymbol{\alpha}_w\right)$.

Let $\widehat{\mathbf{w}}_j = \mathbf{y}_j - \boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \widehat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\widehat{\boldsymbol{\beta}} + \widehat{\mathbf{Z}}_j\widehat{\mathbf{u}}_j$. Estimation of $\boldsymbol{\alpha}$ and σ^2 can now be accomplished by maximizing the log likelihood corresponding to the approximate marginal distribution in (25),

LME step:
$$l_{\mathrm{LB}}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2}\sum_{j=1}^{M}\left\{\log|\mathbf{V}_j(\boldsymbol{\alpha})| + \sigma^{-2}\left(\widehat{\mathbf{w}}_j - \widehat{\mathbf{X}}_j\boldsymbol{\beta}\right)'\mathbf{V}_j{}^{-1}(\boldsymbol{\alpha})\left(\widehat{\mathbf{w}}_j - \widehat{\mathbf{X}}_j\boldsymbol{\beta}\right)\right\} \tag{26}$$

where $n = \sum_{j=1}^{M} n_j$.

Alternatively, when the `reml` option is specified, we take an REML approach and maximize

$$l_{\mathrm{LB},R}(\boldsymbol{\alpha}, \sigma^2) = l_{\mathrm{LB}}(\boldsymbol{\alpha}, \widehat{\boldsymbol{\beta}}(\boldsymbol{\alpha}), \sigma^2) - \frac{1}{2}\sum_{j=1}^{M}\log\left|\sigma^{-2}\widehat{\mathbf{X}}_j{}'\mathbf{V}_j{}^{-1}(\boldsymbol{\alpha})\widehat{\mathbf{X}}_j\right| \tag{27}$$

The LME step (step 2) of the alternating algorithm consists of optimizing an LME log likelihood, in which the response vector is given by $\widehat{\mathbf{w}}_j$ and the fixed- and random-effects design matrices are given by $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Z}}_j$, respectively. `lmeopts(iterate(#))` iterations are performed at this step, unless the CC is met. The CC for LME optimization is controlled by `lmeopts(nrtolerance(#))` and one of `lmeopts(ltolerance(#))` or `lmeopts(tolerance(#))`; see *menlmaxopts* for details.

The LME step produces estimates $\widehat{\boldsymbol{\alpha}}$ and $\widehat{\sigma}^2$. (The estimates $\widehat{\boldsymbol{\beta}}$ can also be obtained at this step, but it is generally more computationally efficient to compute them at the PNLS step.) These estimates will now be used in step 1, the PNLS step.

Stopping rules. One PNLS step and one LME step correspond to one iteration of the alternating algorithm. The log likelihood reported by `menl` at each iteration is the log likelihood (26) or, if the `reml` option is specified, (27) from the last iteration of the LME step. `menl` refers to this log likelihood as "linearization log likelihood" because it corresponds to the log likelihood of the LME model, which was the result of the linearization of the NLME model. The algorithm stops when the linearization likelihoods from successive iterations satisfy `ltolerance(#)`, when the parameter estimates from successive iterations satisfy `tolerance(#)`, or if the model does not converge, when the maximum number of iterations in `iterate()` is reached; see *menlmaxopts* for details about maximization options. Because the alternating algorithm does not provide a joint Hessian matrix for all parameters, there is no check for the tolerance of the scaled gradient; thus the convergence cannot be established in its strict sense. The convergence is declared based on the stopping rules described above.

When $\boldsymbol{\Lambda}_j = \boldsymbol{\Lambda}_j\left(\boldsymbol{\beta}, \mathbf{u}_j, \boldsymbol{\alpha}_w\right)$ depends on $\mathbf{u}_j$ and $\boldsymbol{\beta}$, which is the case, for example, with `resvariance(power _yhat)` and `resvariance(exponential _yhat)`), an intermediate step between the PNLS and the LME step is performed to replace the fixed effects and random effects in $\boldsymbol{\Lambda}_j$, or more precisely in the variance function $g(\cdot)$, by their current estimates from the PNLS step. After that, $\boldsymbol{\Lambda}_j\left(\boldsymbol{\alpha}_w; \widehat{\boldsymbol{\beta}}, \widehat{\mathbf{u}}_j\right) = \boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)$ depends only on $\boldsymbol{\alpha}_w$ because both $\mathbf{u}_j$ and $\boldsymbol{\beta}$ are held fixed at their current estimates throughout the LME step.

Efficient methods for computing (26) or (27) are given in chapters 2 and 5 of Pinheiro and Bates (2000). Namely, to simplify the optimization problem, one can express the optimal values of $\boldsymbol{\beta}$ and σ^2 as functions of $\boldsymbol{\alpha}$ (and thus of $\boldsymbol{\Delta}$ and $\boldsymbol{\alpha}_w$) and work with the profiled log likelihood of $\boldsymbol{\alpha}$.

For the PNLS step, the objective function to be minimized is the penalized sum of squares

$$\sum_{j=1}^{M}\left[\left|\left|\left(\boldsymbol{\Lambda}_j'\right)^{-1/2}\left\{\mathbf{y}_j - \boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)\right\}\right|\right|^2 + \left|\left|\boldsymbol{\Delta}\mathbf{u}_j\right|\right|^2\right]$$

By adding "pseudo"-observations to the data, the PNLS problem can be reexpressed as a standard nonlinear least-squares problem. Thus step 1 of the alternating algorithm is sometimes called the "pseudodata step". Define pseudoobservations $\widetilde{\mathbf{y}}_j$ as follows:

$$\widetilde{\mathbf{y}}_j = \begin{bmatrix} \left(\boldsymbol{\Lambda}_j'\right)^{-1/2}\mathbf{y}_j \\ \mathbf{0} \end{bmatrix} \qquad \widetilde{\boldsymbol{\mu}}\left(\boldsymbol{\beta}, \mathbf{u}_j\right) = \begin{bmatrix} \left(\boldsymbol{\Lambda}_j'\right)^{-1/2}\boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right) \\ \boldsymbol{\Delta}\mathbf{u}_j \end{bmatrix}$$

Then, the PNLS step can be rewritten as

$$\sum_{j=1}^{M}\left|\left|\widetilde{\mathbf{y}}_j - \widetilde{\boldsymbol{\mu}}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)\right|\right|^2$$

Hence, for values of $\boldsymbol{\alpha}$ and σ^2 fixed at the current estimates, the estimation of $\boldsymbol{\beta}$ and $\mathbf{u}_j$ in the PNLS step can be regarded as a standard nonlinear least-squares problem. A popular iterative estimation technique for standard nonlinear least-squares is the Gauss–Newton method (see Pinheiro and Bates [2000, chap. 7] for more details).

After the completion of the alternating algorithm, an extra LME iteration is performed, with fixed effects profiled-out of the likelihood, to reparameterize $[\boldsymbol{\alpha}, \log(\sigma)]$ to their natural metric and to compute their standard errors with the delta method. This step is labeled `Computing standard errors:` in the output of `menl`. If you are interested only in standard errors for fixed effects, you can skip this step by specifying the `nostderr` option, in which case standard errors for the random-effects

and within-group error covariance parameters will not be computed and will be shown as missing in the output table. The standard errors for the fixed effects are obtained from the PNLS step, and the standard errors for random-effects parameters are obtained from the LME step.

Inference on the parameters of the NLME model is based on the approximating LME model with log likelihood and restricted log likelihood functions defined in (26) and (27). Therefore, all the inferential machinery available within the context of LME models can be used. For example, under the LME approximation, the distribution of the (restricted) MLE $\widehat{\boldsymbol{\beta}}$ of the fixed effects is

$$\widehat{\boldsymbol{\beta}} \sim N\left\{\boldsymbol{\beta},\, \sigma^2 \left(\sum_{j=1}^{M} \widehat{\mathbf{X}}_j{}' \mathbf{V}_j{}^{-1}(\boldsymbol{\alpha}) \widehat{\mathbf{X}}_j\right)^{-1}\right\}$$

and for random-effects and within-group error parameters is

$$\begin{bmatrix} \widehat{\boldsymbol{\alpha}} \\ \log\widehat{\sigma} \end{bmatrix} \sim N\left\{\begin{bmatrix} \boldsymbol{\alpha} \\ \log\sigma \end{bmatrix}, I^{-1}(\boldsymbol{\alpha}, \sigma)\right\}$$

where

$$I(\boldsymbol{\alpha}, \sigma) = -\begin{bmatrix} \partial^2 l_{\mathrm{LB}_p}/\partial\boldsymbol{\alpha}\partial\boldsymbol{\alpha}' & \partial^2 l_{\mathrm{LB}_p}/\partial\log\sigma\partial\boldsymbol{\alpha}' \\ \partial^2 l_{\mathrm{LB}_p}/\partial\boldsymbol{\alpha}\partial\log\sigma & \partial^2 l_{\mathrm{LB}_p}/\partial^2\log\sigma \end{bmatrix}$$

and $l_{\mathrm{LB}_p} = l_{\mathrm{LB}_p}(\boldsymbol{\alpha}, \sigma)$ is the approximated log likelihood from the LME step with fixed effects profiled out. Because inference is based on the LME approximation of the original NLME model, asymptotic results are technically "approximately asymptotic" and are thus less accurate than the asymptotic inferential results for LME models as described in [ME] **mixed**.

Initial values

The PNLS step requires starting values for $\boldsymbol{\beta}$ and $\mathbf{u}_j$. These are obtained from the EM algorithm; see, for example, Bates and Pinheiro (1998) for details. You can control optimization within the EM algorithm by specifying the `emtolerance()` and `emiterate()` options. You can also supply your own initial values; see *Examples of specifying initial values*. NLME models are often sensitive to initial values, so it is good practice to try different sets of initial values to verify that your results are robust to them.

References

Bates, D. M., and J. C. Pinheiro. 1998. Computational methods for multilevel modelling. In *Technical Memorandum BL0112140-980226-01TM*. Murray Hill, NJ: Bell Labs, Lucent Technologies. http://ect.bell-labs.com/sl/project/nlme/CompMulti.pdf.

Boeckmann, A. J., L. B. Sheiner, and S. L. Beal. [1994] 2011. *NONMEM Users Guide, Part V: Introductory Guide*. San Francisco: Regents of the University of California. https://nonmem.iconplc.com/nonmem720/guides/v.pdf.

Carroll, R. J., and D. Ruppert. 1988. *Transformation and Weighting in Regression*. New York: Chapman & Hall.

Davidian, M., and D. M. Giltinan. 1995. *Nonlinear Models for Repeated Measurement Data*. Boca Raton, FL: Chapman & Hall/CRC.

——. 2003. Nonlinear models for repeated measurement data: An overview and update. *Journal of Agricultural, Biological, and Environmental Statistics* 8: 387–419.

Demidenko, E. 2013. *Mixed Models: Theory and Applications with R*. 2nd ed. Hoboken, NJ: Wiley.

Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley.

Fitzmaurice, G. M., M. Davidian, G. Verbeke, and G. Molenberghs, ed. 2009. *Longitudinal Data Analysis*. Boca Raton, FL: Chapman & Hall/CRC.

Gibaldi, M., and D. Perrier. 1982. *Pharmacokinetics*. 2nd ed, revised and expanded. New York: Dekker.

Hand, D. J., and M. J. Crowder. 1996. *Practical Longitudinal Data Analysis*. Boca Raton, FL: Chapman & Hall.

Harring, J. R., and J. Liu. 2016. A comparison of estimation methods for nonlinear mixed-effects models under model misspecification and data sparseness: A simulation study. *Journal of Modern Applied Statistical Methods* 15(1): Article 27.

Joyner, W. B., and D. M. Boore. 1981. Peak horizontal acceleration and velocity from strong-motion records including records from the 1979 imperial valley, California, earthquake. *Bulletin of the Seismological Society of America* 71: 2011–2038.

Lindstrom, M. J., and D. M. Bates. 1990. Nonlinear mixed effects models for repeated measures data. *Biometrics* 46: 673–687.

Pierson, R. A., and O. J. Ginther. 1987. Follicular population dynamics during the estrous cycle of the mare. *Animal Reproduction Science* 14: 219–231.

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

——. 2000. *Mixed-Effects Models in S and S-PLUS*. New York: Springer.

Plan, E. L., A. Maloney, F. Mentré, M. O. Karlsson, and J. Bertrand. 2012. Performance comparison of various maximum likelihood nonlinear mixed-effects estimation methods for dose–response models. *AAPS Journal* 14: 420–432.

Rabe-Hesketh, S., and A. Skrondal. 2012. *Multilevel and Longitudinal Modeling Using Stata*. 3rd ed. College Station, TX: Stata Press.

Thompson, W. A., Jr. 1962. The problem of negative estimates of variance components. *Annals of Mathematical Statistics* 33: 273–289.

Vonesh, E., and V. M. Chinchilli. 1997. *Linear and Nonlinear Models for the Analysis of Repeated Measurements*. New York: Dekker.

Wolfinger, R. D., and X. Lin. 1997. Two Taylor-series approximation methods for nonlinear mixed models. *Computational Statistics & Data Analysis* 25: 465–490.

Also see

[ME] **menl postestimation** — Postestimation tools for menl

[ME] **meglm** — Multilevel mixed-effects generalized linear model

[ME] **mixed** — Multilevel mixed-effects linear regression

[ME] **me** — Introduction to multilevel mixed-effects models

[R] **nl** — Nonlinear least-squares estimation

[U] 20 Estimation and postestimation commands