Package: TestREnlme (via r-universe)

July 4, 2026

Type Package

Title Nonparametric Tests for Random Effects in Linear and Nonlinear Mixed-Effects Models

Version 0.1.0

Description Provides nonparametric permutation tests for testing all or any subset of random effects in linear and nonlinear mixed-effects models, without requiring normality or other distributional assumptions on random effects or errors. Three distribution-free variance-component estimators are implemented: Variance Least Squares ('VLS'), Method of Moments ('MM'), and Method of Moments with First-Order Approximation ('MMF'). A permutation procedure is used to obtain finite-sample p-values. Plotting functions support data exploration, model evaluation, and communication of results. Methods are described in Uwimpuhwe, Drikvandi, and Blozis (2026) <doi:10.1002/sim.70605>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Depends R (>= 4.1.0)

Imports nlme (>= 3.1-150), Matrix (>= 1.3-0), MASS (>= 7.3-54), mgsub (>= 1.7.3), matrixcalc (>= 1.0-5), nls.multstart (>= 2.0.0), ggplot2 (>= 3.3.5), expm (>= 0.999-6), withr (>= 3.0.2), stats, utils, rlang

Suggests knitr, rmarkdown, ggpubr (>= 0.4.0), testthat (>= 3.0.0), reshape2 (>= 1.4.4)

VignetteBuilder knitr

URL `https://github.com/germaine86/TestREnlme`

BugReports `https://github.com/germaine86/TestREnlme/issues`

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev

Repository https://germaine86.r-universe.dev

Date/Publication 2026-06-27 14:04:05 UTC

RemoteUrl https://github.com/germaine86/testrenlme

RemoteRef HEAD

RemoteSha f977560272c22b8140cc7440301e810604298f29

Contents

`Beta_hat` *Estimate fixed effects by nonlinear least squares*

Description

Estimates the fixed-effects parameter vector by unweighted NLS (first attempt via `nls`), or by weighted NLS via `nlminb` when `weights` is supplied or when the unweighted NLS fails. Used both for initial unweighted estimation and for GLS re-estimation at the end of VLS using the estimated variance components.

Usage

```
Beta_hat(data, Expr, start, weights = NULL, group = NULL, verbose = 1)
```

Arguments

`data`	A `data.frame` containing all model variables.
`Expr`	A two-sided formula specifying the nonlinear model.
`start`	A named numeric vector of starting values. See the `start` argument of `Dmethod` for details, including automatic computation via `nls.multstart::nls_multstart()` when not supplied.
`weights`	Either `NULL` (default, unweighted), a numeric vector of weights, or a named list of per-subject inverse covariance matrices (used for GLS re-estimation after variance components are estimated).
`group`	Character. Name of the grouping variable, required when `weights` is a list.
`verbose`	Integer (0, 1, or 2). Default `1`.

Value

A named numeric vector of estimated fixed effects.

Examples

```
Expr   <- conc ~ Dose * exp(ai2 + ai3 - ai1) *
             (exp(-Time * exp(ai3)) - exp(-Time * exp(ai2))) /
             (exp(ai2) - exp(ai3))
start  <- c(ai1 = -3.22, ai2 = 0.47, ai3 = -2.45)
Beta_hat(as.data.frame(Theoph), Expr, start)
```

`bootstrap_se` *Bootstrap standard errors for variance component estimates*

Description

Computes bootstrap standard errors for the fixed effects $\hat{\beta}$, random effect variance components $\hat{D}_*$, and error variance $\hat{\sigma}^2$ using one of two bootstrap strategies documented in the literature for mixed-effects models.

Usage

```
bootstrap_se(
  Dobj,
  nboot = 200,
  type = c("case", "residual"),
  seed = NULL,
  verbose = 1
)
```

Arguments

`Dobj`	An object of class `"Dmethod"` returned by `Dmethod`. The estimation method (default `"VLS"`) is inherited from `Dobj`.
`nboot`	Positive integer. Number of bootstrap samples. Default `200`.
`type`	Character. Bootstrap strategy: `"case"` Resample subjects with replacement (default). Valid under minimal assumptions. Recommended for general use Thai et al. (2013). `"residual"` Keep subjects fixed, resample marginal residuals $\hat{e}_i = Y_i - \hat{f}_i$ with replacement. Assumes i.i.d. residuals. See Carpenter et al. (2003), Thai et al. (2013).
`seed`	Optional integer seed for reproducibility. Default `NULL`.
`verbose`	Integer (0, 1, or 2). Default `1`.

Value

A list of class `"Dboot"` with components:

`Beta`	Matrix with columns `Estimate` and `SE` for the fixed effects.
`Dhat`	Matrix with columns `Estimate` and `SE` for each variance/covariance component (flattened via row:column naming).
`Sigma2`	Matrix with columns `Estimate` and `SE` for the error variance.
`Boot_Beta`	Matrix of bootstrap fixed-effects estimates (`nboot` x `length(Beta)`).
`Boot_Dhat`	Array of bootstrap variance component matrices (`k` x `k` x `nboot`).
`Boot_Sigma2`	Numeric vector of bootstrap error variance estimates (length `nboot`).
`type`	The bootstrap strategy used.
`nboot`	Number of bootstrap samples requested.
`nfail`	Number of bootstrap samples that failed.

References

Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003). A novel bootstrap procedure for assessing the relationship between class size and achievement. *Journal of the Royal Statistical Society C*, **52**, 431–443.

Thai, H.T., Mentre, F., Holford, N.H.G., Veyrat-Follet, C. and Comets, E. (2013). A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. *Pharmaceutical Statistics*, **12**, 129–140.

Examples

```
d      <- as.data.frame(Theoph)
Expr   <- conc ~ Dose * exp(ai2 + ai3 - ai1) *
            (exp(-Time * exp(ai3)) - exp(-Time * exp(ai2))) /
            (exp(ai2) - exp(ai3))
start  <- c(ai1 = -3.22, ai2 = 0.47, ai3 = -2.45)
random <- c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2", "ai3 ~ B3 + bi3")
```

```
DVLS   <- Dmethod(d, Expr, group = "Subject",
                 random = random, start = start, method = "VLS")
DVLS[c("Dhat", "Sigma2", "Beta")]

BootSE <- bootstrap_se(DVLS, nboot = 20, type = "case", seed = NULL, verbose = 1)
```

`Dhypothesis_test` *Permutation test for random effects in linear and nonlinear mixed-effects models*

Description

Performs a nonparametric permutation test for all random effects or any user-specified subset, using the test statistic of Drikvandi et al. (2013) adapted for nonlinear mixed-effects models in Uwimpuhwe (2026).

Usage

```
Dhypothesis_test(
  data,
  Expr,
  group,
  random,
  start = NULL,
  bi_out = NULL,
  method = c("VLS", "MM", "MMF"),
  Dhatt = NULL,
  Thatt = NULL,
  MM_base_obj = NULL,
  nperm = 200,
  seed = NULL,
  sig_alpha = 0.05,
  kappa_max = 10000,
  RR_catof = "kappa",
  verbose = 1,
  perm_freq = 10
)
```

Arguments

`data`	A `data.frame` containing all model variables.
`Expr`	A two-sided formula specifying the nonlinear model $f_i(a_i, \gamma)$. The left-hand side is the response variable and the right-hand side defines the nonlinear function using the subject-specific parameter names (e.g., `ai1`, `ai2`, `ai3`) that appear in `start` and `random`.

`group`	Character. Name of the grouping variable.
`random`	A character vector of two-sided formula strings, one per parameter in $a_i = A_i\beta + b_i$, mapping each subject-specific parameter (left-hand side, matching `Expr` and `start`) to its fixed-effects expression plus random effect (right-hand side). For example, `c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2","ai3 ~ B3 + bi3")` specifies that `ai1 = B1 + bi1`, `ai2 = B2 + bi2`, `ai3 = B3 + bi3`.
`start`	A named numeric vector of starting values for all parameters in `Expr`. Names must match those used in `Expr` (e.g., `ai1`, `ai2`, `ai3`). If `NULL` (the default), `Dmethod` attempts to compute starting values automatically using `nls.multstart::nls_multstart()`, searching over multiple initial values within a specified range (e.g. ± 10) for each parameter. If this step fails, provide `start` manually, or fit `nls_multstart()` or `nls()` separately with different starting values, search bounds, or optimisation settings, and use the resulting coefficients as starting values.
`bi_out`	Optional character vector naming the random effects to test. If `NULL` (default) all random effects are tested jointly (Algorithm 1). Otherwise, only the named effects are tested conditionally on the rest being retained.
`method`	Character. One of `"VLS"` (default), `"MM"`, or `"MMF"`.
`Dhatt`	Optional pre-computed `"Dmethod"` object. When supplied, variance estimation is skipped. This is the recommended approach when running multiple tests on the same data, since $\hat{D}_*$ only needs to be computed once and can be reused.
`Thatt`	Optional pre-computed observed test statistic from `Tstat`.
`MM_base_obj`	Optional pre-computed `"MM_base"` object. Only used when `method` is `"MM"` or `"MMF"` and `Dhatt` is `NULL`.
`nperm`	Positive integer. Number of permutations B. Default `200`. Use `1000` or more for publication results.
`seed`	Optional integer seed for reproducibility. Default `NULL`.
`sig_alpha`	Significance level for the reject/do-not-reject decision. Default `0.05`.
`kappa_max`	Condition-number threshold for MM/MMF. Default `1e4`.
`RR_catof`	Exclusion criterion passed to `MM_base`. Either `"kappa"` (default), which uses the condition-number threshold `kappa_max`, or a user-specified numeric threshold applied directly.
`verbose`	Integer (0, 1, or 2). Controls message output: `0` Completely silent. `1` Prints summary: subjects used/excluded, observed statistic, p-value, and decision (default). `2` Also prints every `perm_freq` permutation counter.
`perm_freq`	Integer. When `verbose = 2`, print a permutation progress message every `perm_freq` permutations. Default `10`.

Value

An object of class `"Dtest"`, a list with components:

`Decision` Character, `"Reject H0"` or `"Do not reject H0"`.

`pvalue` The empirical permutation p-value.

`Tobs` The observed test statistic T_{obs}.

`Tperm` Numeric vector of length `nperm` containing the permutation statistics $T^{(1)}, \ldots, T^{(B)}$.

`Dhatt` The `Dmethod` object used.

`bi_out` The random effects tested.

`plot` A **ggplot2** histogram of the permutation null distribution with T_{obs} annotated.

References

Uwimpuhwe, G., Drikvandi, R., and Blozis, S.A. (2026). Testing random effects in nonlinear mixed-effects models. *Statistics in Medicine*. doi:10.1002/sim.70605

Uwimpuhwe, G., Drikvandi, R. and Blozis, S. A. (in preparation). TestREnlme: An R Package for Testing Random Effects in Nonlinear Mixed-Effects Models. *Journal of Statistical Software*.

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

Drikvandi, R., Verbeke, G., Khodadadi, A. and Nia, V. P. (2013). Testing multiple variance components in linear mixed-effects models. *Biostatistics*, **14 (1)**, 144–159.

Examples

```
d      <- as.data.frame(Theoph)
Expr   <- conc ~ Dose * exp(ai2 + ai3 - ai1) *
            (exp(-Time * exp(ai3)) - exp(-Time * exp(ai2))) /
            (exp(ai2) - exp(ai3))
start  <- c(ai1 = -3.22, ai2 = 0.47, ai3 = -2.45)
random <- c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2", "ai3 ~ B3 + bi3")
DVLS  <- Dmethod(d, Expr, group = "Subject",
                   random = random, start = start)


## Permutation test (slow -- use nperm = 200 minimum for real use)
H      <- Dhypothesis_test(d, Expr, group = "Subject",
                           random = random, start = start,
                           Dhatt = DVLS, nperm = 20, seed = 1)
H$pvalue
H$plot
```

`Dmethod` *Non-parametric approach to estimate variance components in a linear and nonlinear mixed-effects model*

Description

Computes the scaled variance covariance matrix $\hat{D}_*$ and the error variance $\hat{\sigma}^2$ using one of three nonparametric estimators: Variance Least Squares (`"VLS"`), Method of Moments (`"MM"`), or Method of Moments with First-Order Approximation (`"MMF"`). The method also estimate weighted fixed effects

Usage

```
Dmethod(
  data,
  Expr,
  group,
  random,
  start = NULL,
  method = c("VLS", "MM", "MMF"),
  MM_base_obj = NULL,
  kappa_max = 10000,
  RR_catof = "kappa",
  Beta_nls = NULL,
  verbose = 1,
  is_permuting = FALSE
)
```

Arguments

`data`	A `data.frame` containing all model variables.
`Expr`	A two-sided formula specifying the nonlinear model $f_i(a_i, \gamma)$. The left-hand side is the response variable and the right-hand side defines the nonlinear function using the subject-specific parameter names (e.g., `ai1`, `ai2`, `ai3`) that appear in `start` and `random`.
`group`	Character. Name of the grouping variable in `data`.
`random`	A character vector of two-sided formula strings, one per parameter in $a_i = A_i\beta + b_i$, mapping each subject-specific parameter (left-hand side, matching `Expr` and `start`) to its fixed-effects expression plus random effect (right-hand side). For example, `c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2","ai3 ~ B3 + bi3")` specifies that `ai1 = B1 + bi1`, `ai2 = B2 + bi2`, `ai3 = B3 + bi3`.
`start`	A named numeric vector of starting values for all parameters in `Expr`. Names must match those used in `Expr` (e.g., `ai1`, `ai2`, `ai3`). If `NULL` (the default), `Dmethod` attempts to compute starting values automatically using `nls.multstart::nls_multstart()`, searching over multiple initial values within a specified range (e.g. ± 10) for each parameter. If this step fails, provide `start` manually, or fit `nls_multstart()` or `nls()` separately with different starting values, search bounds, or optimisation settings, and use the resulting coefficients as starting values.
`method`	Character. One of `"VLS"` (default), `"MM"`, or `"MMF"` estimation method.
`MM_base_obj`	An optional pre-computed object of class `"MM_base"` returned by `MM_base`. When `NULL` (default) and `method` is `"MM"` or `"MMF"`, `MM_base()` is called internally. Supplying a pre-computed object avoids repeating the expensive first-stage NLS fits when calling both `"MM"` and `"MMF"` on the same data.
`kappa_max`	Positive numeric. Condition-number threshold for excluding subjects in MM/MMF. Default `1e4`.
`RR_catof`	Exclusion criterion passed to `MM_base`. Either `"kappa"` (default), which uses the condition-number threshold `kappa_max`, or a user-specified numeric threshold applied directly.

`Beta_nls`	Optional named numeric vector of pre-computed fixed-effects estimates. If `NULL` (default), estimated internally via `Beta_hat`.
`verbose`	Integer (0, 1, or 2). `0` = silent; `1` = summary messages (default); `2` = full progress.
`is_permuting`	Logical. Internal flag set to `TRUE` during the permutation procedure, where the GLS-weighted refit of fixed effects is not needed and `Beta_nls` is used instead. Default `FALSE`.

Value

An object of class `"Dmethod"`, a list with components:

`Dhat` The estimated scaled covariance matrix of random effects $\hat{D}_*$.

`Sigma2` The estimated error variance $\hat{\sigma}^2$.

`Beta` The estimated fixed-effects vector $\hat{\beta}$.

`method` The estimation method used.

`R` Per-subject Jacobian matrices $R_i(\hat{\theta}_0)$.

`Ypred` Per-subject fitted values under fixed effects only.

`residuals` Per-subject residuals $\hat{e}_i$.

`IDconvegence` Subject inclusion/exclusion summary (MM/MMF only; `NULL` otherwise).

Additional internal objects are stored as an `"internal"` attribute (`attr(object, "internal")`) and used by package methods. They are not part of the user interface.

References

Uwimpuhwe, G., Drikvandi, R., and Blozis, S.A. (2026). Testing random effects in nonlinear mixed-effects models. *Statistics in Medicine*. doi:10.1002/sim.70605

Uwimpuhwe, G., Drikvandi, R. and Blozis, S. A. (in preparation). TestREnlme: An R Package for Testing Random Effects in Nonlinear Mixed-Effects Models. *Journal of Statistical Software*.

Demidenko, E. (2013). *Mixed Models: Theory and Applications with R* (2nd ed.). John Wiley & Sons

Drikvandi, R., Verbeke, G., Khodadadi, A. and Nia, V. P. (2013). Testing multiple variance components in linear mixed-effects models. *Biostatistics*, **14 (1)**, 144–159.

Examples

```
d      <- as.data.frame(Theoph)
Expr   <- conc ~ Dose * exp(ai2 + ai3 - ai1) *
            (exp(-Time * exp(ai3)) - exp(-Time * exp(ai2))) /
            (exp(ai2) - exp(ai3))
start  <- c(ai1 = -3.22, ai2 = 0.47, ai3 = -2.45)
random <- c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2", "ai3 ~ B3 + bi3")
DVLS   <- Dmethod(d, Expr, group = "Subject",
                  random = random, start = start)
## method defaults to VLS; explicit: method = "VLS"
DVLS[c("Dhat", "Sigma2", "Beta")]
```

`MM_base` *Compute first-stage quantities for MM and MMF estimators*

Description

Performs the first-stage nonlinear least squares (NLS) fit for each subject individually, and computes the per-subject Jacobian matrices $R_i^T R_i$ and condition numbers κ_i. The returned object can be passed to `Dmethod` via the `MM_base_obj` argument to avoid recomputing the first stage when calling both `method = "MM"` and `method = "MMF"` on the same data.

Usage

```
MM_base(
  data,
  Expr,
  group,
  random,
  start = NULL,
  kappa_max = 10000,
  RR_catof = "kappa",
  verbose = 1
)
```

Arguments

`data`	A `data.frame` containing all variables.
`Expr`	A two-sided formula specifying the nonlinear model $f_i(a_i, \gamma)$. The left-hand side is the response variable and the right-hand side defines the nonlinear function using the subject-specific parameter names (e.g., `ai1`, `ai2`, `ai3`) that appear in `start` and `random`.
`group`	Character. Name of the grouping (subject) variable.
`random`	A character vector of two-sided formula strings, one per parameter in $a_i = A_i\beta + b_i$, mapping each subject-specific parameter (left-hand side, matching `Expr` and `start`) to its fixed-effects expression plus random effect (right-hand side). For example, `c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2","ai3 ~ B3 + bi3")` specifies that `ai1 = B1 + bi1`, `ai2 = B2 + bi2`, `ai3 = B3 + bi3`.
`start`	A named numeric vector of starting values for all parameters in `Expr`. Names must match those used in `Expr` (e.g., `ai1`, `ai2`, `ai3`). If `NULL` (the default), `Dmethod` attempts to compute starting values automatically using
`kappa_max`	Positive numeric. Subjects whose per-subject Jacobian condition number exceeds this threshold are excluded from MM/MMF second-stage estimation. Default `1e4`.
`RR_catof`	Character. Exclusion criterion: `"kappa"` (default) uses the condition-number threshold `kappa_max`; or a user-specified numeric threshold on the $R_i^T R_i$.
`verbose`	Integer (0, 1, or 2). `0` = silent; `1` = prints a summary of subjects retained and excluded (default); `2` = additionally prints per-subject convergence messages.

Value

A list of class `"MM_base"` with components:

`data`	The (re-ordered, group-renumbered) `data.frame` used internally.
`group`	Name of the grouping variable.
`start`	The starting values used.
`Beta_nls`	Named numeric vector of pooled (first-pass) fixed-effects estimates from `Beta_hat`.
`random`	The `random` formulas, aligned with `start`.
`re_names`	Named character vector mapping random-effect names to subject-specific parameter names.
`Tmatrix`	Matrix of per-subject scaled covariance terms (one row per subject, `k^2` columns).
`Ti`	List of per-subject $k \times k$ covariance matrices derived from `Tmatrix`.
`ddi`	List of per-subject $k \times k$ second-stage deviation outer-product matrices (eq.~8.20).
`Lamda`	Minimum eigenvalue of the standardised scatter matrix, used to bias-correct the MM/MMF variance estimates.
`sigma2`	Pooled residual variance across retained subjects.
`Beta_GB`	Named numeric vector of non-random-effect fixed-effects estimates.
`m`	Total number of second-stage fixed-effects parameters (`k * ncol(Q)`).
`k`	Number of random-effects parameters.
`id`, `uid`, `nud`	Subject index vector, unique subject IDs, and number of unique subjects.
`IDworks`	Integer vector of subject IDs retained after filtering.
`Q`	Subject-level design matrix.
`Aai`	List with elements `Ai` (per-subject design matrices) and `a0i` (per-subject parameter estimates).
`kappa`	Named numeric vector of per-subject condition numbers.
`IDconvegence`	List with elements `ID_all`, `ID_used`, and `ID_supressed` (itself a list with `highRR` and `Unconverged`).

Examples

```
d      <- as.data.frame(Theoph)
Expr   <- conc ~ Dose * exp(ai2 + ai3 - ai1) *
            (exp(-Time * exp(ai3)) - exp(-Time * exp(ai2))) /
            (exp(ai2) - exp(ai3))
start  <- c(ai1 = -3.22, ai2 = 0.47, ai3 = -2.45)
random <- c("ai1 ~ B1 + bi1", "ai2 ~ B2 + bi2", "ai3 ~ B3 + bi3")
mb <- MM_base(d, Expr, group = "Subject", random = random, start = start)
```

`plot.Dmethod` *Plot method for* `Dmethod` *objects*

Description

By default all relevant diagnostic plots are produced and arranged in a grid. Use the `which` argument to select specific plots.

Usage

```
## S3 method for class 'Dmethod'
plot(x, time, which = NULL, ...)
```

Arguments

`x`	A `"Dmethod"` object.
`time`	Character. Name of the time variable (required).
`which`	Integer vector specifying which plots to show: `1` = individual profiles, `2` = fitted vs observed, `3` = residuals vs fitted (response), `4` = standardised residuals, `5` = condition numbers (MM/MMF only). Default: all applicable plots.
`...`	Additional arguments passed to individual plot functions.

Value

Invisibly returns a list of `ggplot2` objects.

`plot.Dtest` *Plot method for* `Dtest` *objects*

Description

Displays the permutation null distribution histogram with the observed test statistic annotated.

Usage

```
## S3 method for class 'Dtest'
plot(x, ...)
```

Arguments

`x`	A `"Dtest"` object.
`...`	Additional arguments passed to `plot_perm_hist`.

Value

Invisibly returns the `ggplot2` object.

`plot_condition` *Plot per-subject condition numbers*

Description

Displays the condition numbers κ_i from a `"MM_base"` object or the `MM_base_obj` slot of a `"Dmethod"` object, sorted in decreasing order. A horizontal reference line at `kappa_max` marks the exclusion threshold.

Usage

```
plot_condition(Dobj, kappa_max = 10000, log_scale = TRUE)
```

Arguments

`Dobj`	A `"MM_base_obj"` object from `MM_base`, or a `"Dmethod"` object (for MM/MMF methods only).
`kappa_max`	Numeric. Exclusion threshold to highlight on the plot. Default `1e4`.
`log_scale`	Logical. If `TRUE` (default), the y-axis is on the $\log_{10}$ scale.

Value

A `ggplot2` object.

`plot_fitted` *Plot fitted curves overlaid on observed data*

Description

Draws the model-predicted trajectories on top of the observed data for each subject. When a `"Dmethod"` object is supplied, subject-specific EBLUP-adjusted predictions are used; otherwise only the population fixed-effects curve is shown.

Usage

```
plot_fitted(Dobj, time, subjects = NULL, overlay = TRUE, ncol = 4)
```

Arguments

`Dobj`	An object of class `"Dmethod"` returned by `Dmethod`.
`time`	Character. Name of the time variable in `Dobj$data`.
`subjects`	Optional character or integer vector of subject IDs to plot. If `NULL` (default) all subjects are plotted.
`overlay`	Logical. If `TRUE` (default) observed and fitted curves are drawn in the same panel per subject. If `FALSE`, separate facets are used.
`ncol`	Integer. Number of columns for the subject facets. Default `4`.

Value

A `ggplot2` object.

`plot_perm_hist` *Histogram of the permutation null distribution*

Description

Regenerates or customises the permutation histogram stored in a `"Dtest"` object. The observed statistic T_{obs} is shown as a vertical dashed line, and the empirical p-value and rejection decision are annotated on the plot.

Usage

```
plot_perm_hist(
  Htest,
  bins = 30,
  fill = "steelblue",
  line_col = "red",
  title = NULL
)
```

Arguments

`Htest`	An object of class `"Dtest"` returned by `Dhypothesis_test`.
`bins`	Integer. Number of histogram bins. Default `30`.
`fill`	Character. Fill colour for the histogram bars. Default `"steelblue"`.
`line_col`	Character. Colour for the T_{obs} line. Default `"red"`.
`title`	Character. Plot title. If `NULL` a default title is generated.

Value

A `ggplot2` object.

`plot_profiles` *Plot raw individual trajectories*

Description

Produces a spaghetti plot of the raw observed trajectories for all subjects (or a selected subset), with an optional group-level mean profile and subject highlighting.

Usage

```
plot_profiles(
  data,
  group,
  time,
  response,
  subjects = NULL,
  mean_profile = TRUE,
  highlight = NULL,
  title = "Individual profiles"
)
```

Arguments

`data`	A `data.frame` containing the data.
`group`	Character. Name of the grouping variable.
`time`	Character. Name of the time variable.
`response`	Character. Name of the response variable.
`subjects`	Optional character or integer vector of subject IDs to plot. If `NULL` (default) all subjects are plotted.
`mean_profile`	Logical. Whether to superimpose the group-level mean trajectory as a bold line. Default `TRUE`.
`highlight`	Optional character or integer vector of subject IDs to draw in a contrasting colour (e.g., subjects excluded by the condition-number filter).
`title`	Character. Plot title. Default `"Individual profiles"`.

Value

A `ggplot2` object.

Examples

```
d <- as.data.frame(Theoph)
plot_profiles(d, group = "Subject", time = "Time", response = "conc")
```

`plot_residuals` *Plot residuals versus fitted values*

Description

Produces a residuals-versus-fitted-values scatter plot with a horizontal reference line at zero and a loess smoother to aid detection of systematic misfit or heteroscedasticity.

Usage

```
plot_residuals(Dobj, time, type = c("response", "standardised", "subject"))
```

Arguments

`Dobj`	An object of class `"Dmethod"`.
`time`	Character. Name of the time variable.
`type`	Character. Type of residuals: `"response"` (default) for raw residuals $Y_{ij} - \hat{Y}_{ij}$; `"standardised"` for residuals divided by $\hat{\sigma}$; `"subject"` for subject-level mean residuals.

Value

A `ggplot2` object.

`plot_variance` *Compare variance component estimates across methods*

Description

Produces a side-by-side dot plot (or bar chart) of the estimated variance components $\hat{d}_{*jj}$ for each random effect, comparing results from multiple estimation methods.

Usage

```
plot_variance(
  Dhatt_list,
  component = c("diagonal", "full"),
  title = "Variance component estimates"
)
```

Arguments

`Dhatt_list`	A named list of `"Dmethod"` objects, one per method. For example, `list(VLS = DVLS, MM = DMM, MMF = DMMF)`.
`component`	Character. `"diagonal"` (default) plots only the variance terms $\hat{d}_{*jj}$; `"full"` additionally includes the covariance terms.
`title`	Character. Plot title.

Value

A `ggplot2` object.

`SkillAcq` *Skill acquisition data*

Description

Response latency data from a study of quantitative skill acquisition on a learning task (Blozis 2004). Log-transformed response latencies are recorded for $N = 204$ subjects across $J = 11$ trial blocks, stored in wide format. Used to illustrate fitting a nonlinear mixed-effects model with a subject-level covariate (working memory) incorporated through the second-stage model, and to demonstrate reshaping wide-format longitudinal data into the long format required by `Dmethod` and `Dhypothesis_test`.

Usage

```
SkillAcq
```

Format

A data frame with 204 rows (one per subject) and 13 columns:

id Subject identifier.

ly1 Log-transformed response latency at trial block 1.

ly2 Log-transformed response latency at trial block 2.

ly3 Log-transformed response latency at trial block 3.

ly4 Log-transformed response latency at trial block 4.

ly5 Log-transformed response latency at trial block 5.

ly6 Log-transformed response latency at trial block 6.

ly7 Log-transformed response latency at trial block 7.

ly8 Log-transformed response latency at trial block 8.

ly9 Log-transformed response latency at trial block 9.

ly10 Log-transformed response latency at trial block 10.

ly11 Log-transformed response latency at trial block 11.

wm2 Subject-level working-memory covariate.

Details

`SkillAcq` is stored in wide format, as commonly encountered in practice. Before use with `Dmethod` it must be reshaped to long format, with one row per subject-trial observation; see **Examples** below.

The nonlinear mixed-effects model used for this dataset is

$$Y_{ij} = a_{i1} - (a_{i1} + a_{i0}) \exp(a_{i2} T_{ij}) + \varepsilon_{ij},$$

with

$$a_{ik} = \beta_{0k} + \beta_{1k} wm2_i + b_{ik}, \quad k \in \{0, 1, 2\},$$

where a_{i0}, a_{i1}, and a_{i2} represent, respectively, the subject-specific initial performance offset, lower asymptote, and learning rate, each with a regression component on `wm2` and a subject-specific random effect. The questions of interest are whether all three random effects are necessary, and whether one or more can be removed to obtain a more parsimonious model; see `Dhypothesis_test`.

Source

Blozis, S. A. (2004). Structured latent curve models for the study of change in multivariate repeated measures. *Psychological Methods*, **9(3)**, 334–353. https://doi.org/10.1037/1082-989X.9.3.334

Used as a worked example in Uwimpuhwe, G., Drikvandi, R. and Blozis, S. A. (in preparation). TestREnlme: An R Package for Testing Random Effects in Nonlinear Mixed-Effects Models. *Journal of Statistical Software*.

Examples

```
## Reshape from wide (ly1..ly11) to long format
qrt  <- data.frame(SkillAcq)
qrt1 <- reshape2::melt(qrt, id.vars = c("id", "wm2"),
                       variable.name = "ly", value.name = "Y")
qrt1$t <- as.numeric(sub("ly", "", qrt1$ly))

## Model: Y_ij = ai1 - (ai1 + ai0) * exp(ai2 * t_ij) + e_ij
## with aik = B0k + B1k * wm2_i + bik, k in {0, 1, 2}
Expr_learn <- Y ~ ai1 - (ai1 + ai0) * exp(ai2 * t)
random_learn <- c("ai0 ~ B00 + B10 * wm2 + bi0",
                  "ai1 ~ B01 + B11 * wm2 + bi1",
                  "ai2 ~ B02 + B12 * wm2 + bi2")


## Estimate variance components (VLS) and test all three random effects
DVLS_learn <- Dmethod(qrt1, Expr_learn, group = "id",
                      random = random_learn, start = NULL)
DVLS_learn[c("Dhat", "Sigma2", "Beta")]

H_learn <- Dhypothesis_test(qrt1, Expr_learn, group = "id",
                            random = random_learn, start = NULL,
                            Dhatt = DVLS_learn, nperm = 200, seed = 1)
H_learn$pvalue
```

`Tstat` *Compute the observed test statistic*

Description

Computes $T_{\text{obs}} = N^{-1} \sum_i \text{tr}(R_i \hat{D}_* R_i^T)$ from a fitted `Dmethod` object.

Usage

```
Tstat(Dobj, bi_out = NULL)
```

Arguments

`Dobj`	An object of class `"Dmethod"` returned by `Dmethod`.
`bi_out`	Optional character vector of random-effect names to test. If `NULL` (default) all random effects are included.

Value

A scalar, the value of T_{obs}.

`ZandYPred` *Compute Jacobian matrix and fixed-effects predicted values*

Description

Evaluates the Jacobian $R_i(\hat{\theta}_0)$ of the nonlinear model function with respect to the random-effects parameters, and computes fitted values under the fixed-effects-only model.

Usage

```
ZandYPred(data, Expr, group, random, Beta_nls = NULL, start = NULL)
```

Arguments

`data`	A `data.frame`.
`Expr`	A two-sided formula for the nonlinear model.
`group`	Character name of the grouping variable.
`random`	A named list or vector of one-sided formulas mapping each parameter to its random-effect expression, e.g. `c(B1 ~ B1 + bi1, B2 ~ B2 + bi2)`.
`Beta_nls`	Named numeric vector of fixed-effects estimates from `Beta_hat`.
`start`	A named numeric vector of starting values. See the `start` argument of `Dmethod` for details, including automatic computation via `nls.multstart::nls_multstart()` when not supplied.

Value

A list with components:

`R` Named list of per-subject Jacobian matrices $R_i(\hat{\theta}_0)$.

`Ypred` Named list of per-subject predicted vectors $f_i(A_i\hat{\beta}, \hat{\gamma})$.

`residuals` Named list of per-subject residual vectors $\hat{e}_i = Y_i - f_i$.

Index