Package 'mmrm'

December 9, 2025

Type Package

Title Mixed Models for Repeated Measures

Version 0.3.16

Description Mixed models for repeated measures (MMRM) are a popular choice for analyzing longitudinal continuous outcomes in randomized clinical trials and beyond; see Cnaan, Laird and Slasor (1997) <doi:10.1002/(SICI)1097-0258(19971030)16:20%3C2349::AID-SIM667%3E3.0.CO;2-E> for a tutorial and Mallinckrodt, Lane, Schnell, Peng and Mancuso (2008) <doi:10.1177/009286150804200402> for a review. This package implements MMRM based on the marginal linear model without random effects using Template Model Builder ('TMB') which enables fast and robust model fitting. Users can specify a variety of covariance matrices, weight observations, fit models with restricted or standard maximum likelihood inference, perform hypothesis testing with Satterthwaite or Kenward-Roger adjustment, and extract least square means estimates by using 'emmeans'.

License Apache License 2.0

URL https://openpharma.github.io/mmrm/

BugReports https://github.com/openpharma/mmrm/issues

Depends R (>= 4.1)

Imports checkmate (>= 2.0), generics, lifecycle, MASS, Matrix, methods, nlme, parallel, Rcpp, Rdpack, stats, stringr, tibble, TMB (>= 1.9.1), utils

Suggests broom, broom.helpers, car (>= 3.1.2), cli, clubSandwich, clusterGeneration, dplyr, emmeans (>= 1.6), estimability, ggplot2, glmmTMB, hardhat, knitr, lme4, lmerTest, microbenchmark, mockery, parallelly (>= 1.32.0), parsnip (>= 1.1.0), purrr, rmarkdown, sasr, scales, testthat (>= 3.0.0), tidymodels, withr, xml2

LinkingTo Rcpp, RcppEigen, testthat, TMB (>= 1.9.1)

VignetteBuilder knitr

RdMacros Rdpack

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation yes

RoxygenNote 7.3.3

Collate 'between-within.R' 'catch-routine-registration.R'
'component.R' 'cov_struct.R' 'data.R' 'empirical.R' 'fit.R'
'kenwardroger.R' 'mmrm-methods.R' 'mmrm-package.R' 'utils.R'
'residual.R' 'utils-nse.R' 'utils-formula.R' 'satterthwaite.R'
'skipping.R' 'tidiers.R' 'testing.R' 'tmb-methods.R' 'tmb.R'
'interop-emmeans.R' 'interop-parsnip.R' 'interop-car.R' 'zzz.R'

Author Daniel Sabanes Bove [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0176-9239>),
Liming Li [aut] (ORCID: <https://orcid.org/0009-0008-6870-0878>),
Julia Dedic [aut],
Doug Kelkhoff [aut],
Kevin Kunzmann [aut],
Brian Matthew Lang [aut],
Christian Stock [aut],
Ya Wang [aut],
Craig Gower-Page [ctb],
Dan James [aut],
Jonathan Sidi [aut],
Daniel Leibovitz [aut],
Daniel D. Sjoberg [aut] (ORCID:
<https://orcid.org/0000-0003-0862-2018>),
Nikolas Ivan Krieger [aut] (ORCID:
<https://orcid.org/0000-0002-4581-3545>),
Lukas A. Widmer [ctb] (ORCID: <https://orcid.org/0000-0003-1471-3493>),
Boehringer Ingelheim Ltd. [cph, fnd],
Gilead Sciences, Inc. [cph, fnd],
F. Hoffmann-La Roche AG [cph, fnd],
Merck Sharp & Dohme, Inc. [cph, fnd],
AstraZeneca plc [cph, fnd],
inferential.biostatistics GmbH [cph, fnd]

Maintainer Daniel Sabanes Bove <daniel.sabanes_bove@rconis.com>

Repository CRAN

Date/Publication 2025-12-09 15:20:02 UTC

Contents

`mmrm-package` `mmrm` *Package*

Description

`mmrm` implements mixed models with repeated measures (MMRM) in R.

Author(s)

Maintainer: Daniel Sabanes Bove <daniel.sabanes_bove@rconis.com> (ORCID)

Authors:

- Liming Li <liming.li1@astrazeneca.com> (ORCID)
- Julia Dedic <julia.dedic@roche.com>
- Doug Kelkhoff <doug.kelkhoff@roche.com>
- Kevin Kunzmann <kevin.kunzmann@boehringer-ingelheim.com>
- Brian Matthew Lang <brian.lang@msd.com>
- Christian Stock <christian.stock@boehringer-ingelheim.com>
- Ya Wang <ya.wang10@gilead.com>
- Dan James <dan.james@astrazeneca.com>
- Jonathan Sidi <yoni@pinpointstrategies.io>
- Daniel Leibovitz <daniel.leibovitz@roche.com>
- Daniel D. Sjoberg <sjobergd@gene.com> (ORCID)

- Nikolas Ivan Krieger <nikolas.krieger@experis.com> (ORCID)

Other contributors:

- Craig Gower-Page <craig.gower-page@roche.com> [contributor]
- Lukas A. Widmer (ORCID) [contributor]

See Also

Useful links:

- https://openpharma.github.io/mmrm/
- Report bugs at https://github.com/openpharma/mmrm/issues

`as.cov_struct` *Coerce into a Covariance Structure Definition*

Description

[Stable]

Usage

```
as.cov_struct(x, ...)

## S3 method for class 'formula'
as.cov_struct(x, warn_partial = TRUE, ...)
```

Arguments

`x`	an object from which to derive a covariance structure. See object specific sections for details.
`...`	additional arguments unused.
`warn_partial`	(`flag`) whether to emit a warning when parts of the formula are disregarded.

Details

A covariance structure can be parsed from a model definition formula or call. Generally, covariance structures defined using non-standard evaluation take the following form:

```
type( (visit, )* visit | (group /)? subject )
```

For example, formulas may include terms such as

```
us(time | subject)
cp(time | group / subject)
sp_exp(coord1, coord2 | group / subject)
```

Note that only `sp_exp` (spatial) covariance structures may provide multiple coordinates, which identify the Euclidean distance between the time points.

Value

A `cov_struct()` object.

Methods (by class)

- `as.cov_struct(formula)`: When provided a formula, any specialized functions are assumed to be covariance structure definitions and must follow the form:

  ```
  y ~ xs + type( (visit, )* visit | (group /)? subject )
  ```

  Any component on the right hand side of a formula is considered when searching for a covariance definition.

See Also

Other covariance types: `cov_struct()`, `covariance_types`

Examples

```
# provide a covariance structure as a right-sided formula
as.cov_struct(~ csh(visit | group / subject))

# when part of a full formula, suppress warnings using `warn_partial = FALSE`
as.cov_struct(y ~ x + csh(visit | group / subject), warn_partial = FALSE)
```

bcva_data *Example Data on BCVA*

Description

[Stable]

Usage

```
bcva_data
```

Format

A `tibble` with 10,000 rows and 7 variables:

- `USUBJID`: subject ID.
- `VISITN`: visit number (numeric).
- `AVISIT`: visit number (factor).
- `ARMCD`: treatment, `TRT` or `CTL`.
- `RACE`: 3-category race.
- `BCVA_BL`: BCVA at baseline.
- `BCVA_CHG`: Change in BCVA at study visits.

Note

Measurements of BCVA (best corrected visual acuity) is a measure of how how many letters a person can read off of an eye chart using corrective lenses or contacts. This a common endpoint in ophthalmology trials.

Source

This is an artificial dataset.

component *Component Access for* `mmrm_tmb` *Objects*

Description

[Stable]

Usage

```
component(
  object,
 name = c("cov_type", "subject_var", "n_theta", "n_subjects", "n_timepoints", "n_obs",
    "beta_vcov", "beta_vcov_complete", "varcor", "score_per_subject", "formula",
   "dataset", "n_groups", "reml", "convergence", "evaluations", "method", "optimizer",
   "conv_message", "call", "theta_est", "beta_est", "beta_est_complete", "beta_aliased",
   "x_matrix", "x_matrix_complete", "y_vector", "neg_log_lik", "jac_list", "theta_vcov",
    "full_frame", "xlev", "contrasts")
)
```

Arguments

object	(mmrm_tmb) the fitted MMRM.
name	(character) the component(s) to be retrieved.

Details

Available component() names are as follows:

- call: low-level function call which generated the model.
- formula: model formula.
- dataset: data set name.
- cov_type: covariance structure type.
- n_theta: number of parameters.
- n_subjects: number of subjects.
- n_timepoints: number of modeled time points.
- n_obs: total number of observations.
- reml: was REML used (ML was used if FALSE).
- neg_log_lik: negative log likelihood.
- convergence: convergence code from optimizer.
- conv_message: message accompanying the convergence code.
- evaluations: number of function evaluations for optimization.
- method: Adjustment method which was used (for mmrm objects), otherwise NULL (for mmrm_tmb objects).
- beta_vcov: estimated variance-covariance matrix of coefficients (excluding aliased coefficients). When Kenward-Roger/Empirical adjusted coefficients covariance matrix is used, the adjusted covariance matrix is returned (to still obtain the original asymptotic covariance matrix use object$beta_vcov).
- beta_vcov_complete: estimated variance-covariance matrix including aliased coefficients with entries set to NA.

- `varcor`: estimated covariance matrix for residuals. If there are multiple groups, a named list of estimated covariance matrices for residuals will be returned. The names are the group levels.
- `score_per_subject`: score per subject in empirical covariance. See the vignette `vignette("coef_vcov", package = "mmrm")`.
- `theta_est`: estimated variance parameters.
- `beta_est`: estimated coefficients (excluding aliased coefficients).
- `beta_est_complete`: estimated coefficients including aliased coefficients set to `NA`.
- `beta_aliased`: whether each coefficient was aliased (i.e. cannot be estimated) or not.
- `theta_vcov`: estimated variance-covariance matrix of variance parameters.
- `x_matrix`: design matrix used (excluding aliased columns).
- `x_matrix_complete`: design matrix used, including aliased columns.
- `xlev`: a named list of character vectors giving the full set of levels to be assumed for each factor.
- `contrasts`: a list of contrasts used for each factor.
- `y_vector`: response vector used.
- `jac_list`: Jacobian, see `h_jac_list()` for details.
- `full_frame`: `data.frame` with `n` rows containing all variables needed in the model.

Value

The corresponding component of the object, see details.

See Also

In the `lme4` package there is a similar function `getME()`.

Examples

```
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
# Get all available components.
component(fit)
# Get convergence code and message.
component(fit, c("convergence", "conv_message"))
# Get modeled formula as a string.
component(fit, c("formula"))
```

`covariance_types` *Covariance Types*

Description

[Stable]

Usage

```
cov_types(
  form = c("name", "abbr", "habbr"),
  filter = c("heterogeneous", "spatial")
)
```

Arguments

`form`	(`character`) covariance structure type name form. One or more of `"name"`, `"abbr"` (abbreviation), or `"habbr"` (heterogeneous abbreviation).
`filter`	(`character`) covariance structure type filter. One or more of `"heterogeneous"` or `"spatial"`.

Value

A character vector of accepted covariance structure type names and abbreviations.

Abbreviations for Covariance Structures

Common Covariance Structures::

Structure	Description	Parameters	(i, j) element
ad	Ante-dependence	m	$\sigma^2 \prod_{k=i}^{j-1} \rho_k$
adh	Heterogeneous ante-dependence	$2m-1$	$\sigma_i \sigma_j \prod_{k=i}^{j-1} \rho_k$
ar1	First-order auto-regressive	2	$\sigma^2 \rho^{\lvert i-j \rvert}$
ar1h	Heterogeneous first-order auto-regressive	$m+1$	$\sigma_i \sigma_j \rho^{\lvert i-j \rvert}$
cs	Compound symmetry	2	$\sigma^2 [\rho I(i \neq j) + I(i = j)]$
csh	Heterogeneous compound symmetry	$m+1$	$\sigma_i \sigma_j [\rho I(i \neq j) + I(i = j)]$
toep	Toeplitz	m	$\sigma_{\lvert i-j \rvert + 1}$
toeph	Heterogeneous Toeplitz	$2m-1$	$\sigma_i \sigma_j \rho_{\lvert i-j \rvert}$
us	Unstructured	$m(m+1)/2$	σ_{ij}

where i and j denote i-th and j-th time points, respectively, out of total m time points, $1 \leq i, j \leq m$.

Note

The **ante-dependence** covariance structure in this package refers to homogeneous ante-dependence, while the ante-dependence covariance structure from SAS `PROC MIXED` refers to heterogeneous ante-dependence and the homogeneous version is not available in SAS.

For all non-spatial covariance structures, the time variable must be coded as a factor.

Spatial Covariance structures::

Structure	Description	Parameters	(i, j) element
sp_exp	spatial exponential	2	$\sigma^2 \rho^{-d_{ij}}$

where d_{ij} denotes the Euclidean distance between time points i and j.

See Also

Other covariance types: `as.cov_struct()`, `cov_struct()`

`cov_struct` *Define a Covariance Structure*

Description

[Stable]

Usage

```
cov_struct(
  type = cov_types(),
  visits,
  subject,
  group = character(),
  heterogeneous = FALSE
)
```

Arguments

`type`	(`string`) the name of the covariance structure type to use. For available options, see `cov_types()`. If a type abbreviation is used that implies heterogeneity (e.g. `cph`) and no value is provided to `heterogeneous`, then the heterogeneity is derived from the type name.
`visits`	(`character`) a vector of variable names to use for the longitudinal terms of the covariance structure. Multiple terms are only permitted for the `"spatial"` covariance type.
`subject`	(`string`) the name of the variable that encodes a subject identifier.

group (string)
optionally, the name of the variable that encodes a grouping variable for subjects.

heterogeneous (flag)

Value

A cov_struct object.

See Also

Other covariance types: as.cov_struct(), covariance_types

Examples

```
cov_struct("csh", "AVISITN", "USUBJID")
cov_struct("spatial", c("VISITA", "VISITB"), group = "GRP", subject = "SBJ")
```

---

df_1d *Calculation of Degrees of Freedom for One-Dimensional Contrast*

Description

[Stable] Calculates the estimate, adjusted standard error, degrees of freedom, t statistic and p-value for one-dimensional contrast.

Usage

```
df_1d(object, contrast)
```

Arguments

object (mmrm)
the MMRM fit.

contrast (numeric)
contrast vector. Note that this should not include elements for singular coefficient estimates, i.e. only refer to the actually estimated coefficients.

Value

List with est, se, df, t_stat and p_val.

Examples

```
object <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
contrast <- numeric(length(object$beta_est))
contrast[3] <- 1
df_1d(object, contrast)
```

---

df_md *Calculation of Degrees of Freedom for Multi-Dimensional Contrast*

Description

[Stable] Calculates the estimate, standard error, degrees of freedom, t statistic and p-value for m-dimensional contrast, depending on the method used in `mmrm()`.

Usage

```
df_md(object, contrast)
```

Arguments

`object`	(`mmrm`) the MMRM fit.
`contrast`	(`matrix`) numeric contrast matrix, if given a `numeric` then this is coerced to a row vector. Note that this should not include elements for singular coefficient estimates, i.e. only refer to the actually estimated coefficients.

Value

List with `num_df`, `denom_df`, `f_stat` and `p_val` (2-sided p-value).

Examples

```
object <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
contrast <- matrix(data = 0, nrow = 2, ncol = length(object$beta_est))
contrast[1, 2] <- contrast[2, 3] <- 1
df_md(object, contrast)
```

emmeans_support *Support for* emmeans

Description

[Stable]

This package includes methods that allow mmrm objects to be used with the emmeans package. emmeans computes estimated marginal means (also called least-square means) for the coefficients of the MMRM. We can also e.g. obtain differences between groups by applying pairs() on the object returned by emmeans::emmeans().

Examples

```
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
if (require(emmeans)) {
  emmeans(fit, ~ ARMCD | AVISIT)
  pairs(emmeans(fit, ~ ARMCD | AVISIT), reverse = TRUE)
}
```

emp_start *Empirical Starting Value*

Description

Obtain empirical start value for unstructured covariance

Usage

```
emp_start(
  y_vector,
  x_matrix,
  full_frame,
  visit_var,
  subject_var,
  subject_groups,
  ...
)
```

Arguments

y_vector	(numeric) response variable.
x_matrix	(matrix) design matrix.
full_frame	(data.frame) full data frame used for model fitting.
visit_var	(string) visit variable.
subject_var	(string) subject id variable.
subject_groups	(factor) subject group assignment.
...	not used.

Details

This emp_start only works for unstructured covariance structure. It uses linear regression to first obtain the coefficients and use the residuals to obtain the empirical variance-covariance, and it is then used to obtain the starting values.

Value

A numeric vector of starting values.

---

fev_data *Example Data on FEV1*

Description

[Stable]

Usage

```
fev_data
```

Format

A tibble with 800 rows and 7 variables:

- USUBJID: subject ID.
- AVISIT: visit number.
- ARMCD: treatment, TRT or PBO.
- RACE: 3-category race.
- SEX: sex.
- FEV1_BL: FEV1 at baseline (%).
- FEV1: FEV1 at study visits.

- `WEIGHT`: weighting variable.
- `VISITN`: integer order of the visit.
- `VISITN2`: coordinates of the visit for distance calculation.

Note

Measurements of FEV1 (forced expired volume in one second) is a measure of how quickly the lungs can be emptied. Low levels of FEV1 may indicate chronic obstructive pulmonary disease (COPD).

Source

This is an artificial dataset.

---

fit_mmrm *Low-Level Fitting Function for MMRM*

Description

[Stable]

This is the low-level function to fit an MMRM. Note that this does not try different optimizers or adds Jacobian information etc. in contrast to `mmrm()`.

Usage

```
fit_mmrm(
  formula,
  data,
  weights,
  reml = TRUE,
  covariance = NULL,
  tmb_data,
  formula_parts,
  control = mmrm_control()
)
```

Arguments

`formula`	(`formula`) model formula with exactly one special term specifying the visits within subjects, see details.
`data`	(`data.frame`) input data containing the variables used in `formula`.
`weights`	(`vector`) input vector containing the weights.

reml	(flag) whether restricted maximum likelihood (REML) estimation is used, otherwise maximum likelihood (ML) is used.
covariance	(cov_struct) A covariance structure type definition, or value that can be coerced to a covariance structure using as.cov_struct(). If no value is provided, a structure is derived from the provided formula.
tmb_data	(mmrm_tmb_data) object.
formula_parts	(mmrm_tmb_formula_parts) list with formula parts from h_mmrm_tmb_formula_parts().
control	(mmrm_control) list of control options produced by mmrm_control().

Details

The formula typically looks like:

```
FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID)
```

which specifies response and covariates as usual, and exactly one special term defines which covariance structure is used and what are the visit and subject variables.

Always use only the first optimizer if multiple optimizers are provided.

Value

List of class mmrm_tmb, see h_mmrm_tmb_fit() for details. In addition, it contains elements call and optimizer.

Examples

```
formula <- FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID)
data <- fev_data
system.time(result <- fit_mmrm(formula, data, rep(1, nrow(fev_data))))
```

---

fit_single_optimizer *Fitting an MMRM with Single Optimizer*

Description

[Stable]

This function helps to fit an MMRM using TMB with a single optimizer, while capturing messages and warnings.

Usage

```
fit_single_optimizer(
  formula,
  data,
  weights,
  reml = TRUE,
  covariance = NULL,
  tmb_data,
  formula_parts,
  ...,
  control = mmrm_control(...)
)
```

Arguments

formula	(formula) the model formula, see details.
data	(data) the data to be used for the model.
weights	(vector) an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
reml	(flag) whether restricted maximum likelihood (REML) estimation is used, otherwise maximum likelihood (ML) is used.
covariance	(cov_struct) a covariance structure type definition as produced with cov_struct(), or value that can be coerced to a covariance structure using as.cov_struct(). If no value is provided, a structure is derived from the provided formula.
tmb_data	(mmrm_tmb_data) object.
formula_parts	(mmrm_tmb_formula_parts) object.
...	Additional arguments to pass to mmrm_control().
control	(mmrm_control) object.

Details

fit_single_optimizer will fit the mmrm model using the control provided. If there are multiple optimizers provided in control, only the first optimizer will be used. If tmb_data and formula_parts are both provided, formula, data, weights, reml, and covariance are ignored.

Value

The mmrm_fit object, with additional attributes containing warnings, messages, optimizer used and convergence status in addition to the mmrm_tmb contents.

Examples

```
mod_fit <- fit_single_optimizer(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data,
  weights = rep(1, nrow(fev_data)),
  optimizer = "nlminb"
)
attr(mod_fit, "converged")
```

`format.cov_struct` *Format Covariance Structure Object*

Description

Format Covariance Structure Object

Usage

```
## S3 method for class 'cov_struct'
format(x, ...)
```

Arguments

`x`	(`cov_struct`) a covariance structure object.
`...`	Additional arguments unused.

Value

A formatted string for `x`.

`mmrm` *Fit an MMRM*

Description

[Stable]

This is the main function fitting the MMRM.

Usage

```
mmrm(
  formula,
  data,
  weights = NULL,
  covariance = NULL,
  reml = TRUE,
  control = mmrm_control(...),
  ...
)
```

Arguments

formula	(formula) the model formula, see details.
data	(data) the data to be used for the model.
weights	(vector) an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
covariance	(cov_struct) a covariance structure type definition as produced with cov_struct(), or value that can be coerced to a covariance structure using as.cov_struct(). If no value is provided, a structure is derived from the provided formula.
reml	(flag) whether restricted maximum likelihood (REML) estimation is used, otherwise maximum likelihood (ML) is used.
control	(mmrm_control) fine-grained fitting specifications list created with mmrm_control().
...	arguments passed to mmrm_control().

Details

The formula typically looks like: FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID) so specifies response and covariates as usual, and exactly one special term defines which covariance structure is used and what are the time point and subject variables. The covariance structures in the formula can be found in covariance_types.

The time points have to be unique for each subject. That is, there cannot be time points with multiple observations for any subject. The rationale is that these observations would need to be correlated, but it is not possible within the currently implemented covariance structure framework to do that correctly. Moreover, for non-spatial covariance structures, the time variable must be a factor variable.

When optimizer is not set, first the default optimizer (L-BFGS-B) is used to fit the model. If that converges, this is returned. If not, the other available optimizers from h_get_optimizers(), including BFGS, CG and nlminb are tried (in parallel if n_cores is set and not on Windows). If none of the optimizers converge, then the function fails. Otherwise the best fit is returned.

Note that fine-grained control specifications can either be passed directly to the mmrm function, or via the control argument for bundling together with the mmrm_control() function. Both cannot be used together, since this would delete the arguments passed via mmrm.

Value

An mmrm object.

Note

The mmrm object is also an mmrm_fit and an mmrm_tmb object, therefore corresponding methods also work (see mmrm_tmb_methods).

Additional contents depend on vcov (see mmrm_control()):

- If Kenward-Roger covariance matrix is used, kr_comp contains necessary components and beta_vcov_adj includes the adjusted coefficients covariance matrix.
- If Empirical covariance matrix is used, beta_vcov_adj contains the corresponding coefficients covariance matrix estimate. In addition, empirical_g_mat contains the empirical g matrix, which is used to calculate the Satterthwaite degrees of freedom. The score_per_subject contains the empirical score per subject.
- If Asymptotic covariance matrix is used in combination with Satterthwaite d.f. adjustment, the Jacobian information jac_list is included.

Note that these additional elements might change over time and are to be considered internal implementation details rather than part of the public user interface of the package.

Use of the package emmeans is supported, see emmeans_support.

NA values are always omitted regardless of na.action setting.

When the number of visit levels is large, it usually requires large memory to create the covariance matrix. By default, the maximum allowed visit levels is 100, and if there are more visit levels, a confirmation is needed if run interactively. You can use options(mmrm.max_visits = <target>) to increase the maximum allowed number of visit levels. In non-interactive sessions the confirmation is not raised and will directly give you an error if the number of visit levels exceeds the maximum.

Examples

```
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)

# Direct specification of control details:
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data,
  weights = fev_data$WEIGHTS,
  method = "Kenward-Roger"
)

# Alternative specification via control argument (but you cannot mix the
```

```
# two approaches):
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data,
  control = mmrm_control(method = "Kenward-Roger")
)
```

`mmrm_control` *Control Parameters for Fitting an MMRM*

Description

[Stable] Fine-grained specification of the MMRM fit details is possible using this control function.

Usage

```
mmrm_control(
  n_cores = 1L,
  method = c("Satterthwaite", "Kenward-Roger", "Residual", "Between-Within"),
  vcov = NULL,
  start = std_start,
  accept_singular = TRUE,
  drop_visit_levels = TRUE,
  disable_theta_vcov = FALSE,
  ...,
  optimizers = h_get_optimizers(...)
)
```

Arguments

`n_cores` (`count`)
number of cores to be used.

`method` (`string`)
adjustment method for degrees of freedom.

`vcov` (`string`)
coefficients covariance matrix adjustment method.

`start` (`NULL`, `numeric` or `function`)
optional start values for variance parameters. See details for more information.

`accept_singular` (`flag`)
whether singular design matrices are reduced to full rank automatically and additional coefficient estimates will be missing.

`drop_visit_levels` (`flag`)
whether to drop levels for visit variable, if visit variable is a factor, see details.

disable_theta_vcov
: (flag)
whether to disable calculation of variance-covariance matrix for variance parameters. This can speed up fitting when there are many variance parameters, see details.

...
: additional arguments passed to h_get_optimizers().

optimizers
: (list)
optimizer specification, created with h_get_optimizers(). Please note that optimizers using the Hessian will not be compatible with disable_theta_vcov = TRUE and an error will be raised in that case.

Details

For example, if the data only has observations at visits VIS1, VIS3 and VIS4, by default they are treated to be equally spaced, the distance from VIS1 to VIS3, and from VIS3 to VIS4, are identical. However, you can manually convert this visit into a factor, with levels = c("VIS1", "VIS2", "VIS3", "VIS4"), and also use drop_visits_levels = FALSE, then the distance from VIS1 to VIS3 will be double, as VIS2 is a valid visit. However, please be cautious because this can lead to convergence failure when using an unstructured covariance matrix and there are no observations at the missing visits.

- The method and vcov arguments specify the degrees of freedom and coefficients covariance matrix adjustment methods, respectively.
  - Allowed vcov includes: "Asymptotic", "Kenward-Roger", "Kenward-Roger-Linear", "Empirical" (CR0), "Empirical-Jackknife" (CR3), and "Empirical-Bias-Reduced" (CR2).
  - Allowed method includes: "Satterthwaite", "Kenward-Roger", "Between-Within" and "Residual".
  - If method is "Kenward-Roger" then only "Kenward-Roger" or "Kenward-Roger-Linear" are allowed for vcov.
- The vcov argument can be NULL to use the default covariance method depending on the method used for degrees of freedom, see the following table:

method	Default vcov
Satterthwaite	Asymptotic
Kenward-Roger	Kenward-Roger
Residual	Empirical
Between-Within	Asymptotic

- Please note that "Kenward-Roger" for "Unstructured" covariance gives different results compared to SAS; Use "Kenward-Roger-Linear" for vcov instead for better matching of the SAS results.
- The argument start is used to facilitate the choice of initial values for fitting the model. If function is provided, make sure its parameter is a valid element of mmrm_tmb_data or mmrm_tmb_formula_parts and it returns a numeric vector. By default or if NULL is provided, std_start will be used. Other implemented methods include emp_start.

- Disabling the theta_vcov calculation can speed up the fitting process when there are many variance parameters. However, this has also drawbacks. We can no longer check that the variance parameter estimates are indeed at a local maximum of the log-likelihood surface, and therefore convergence issues might go unnoticed. In addition, some covariance matrix adjustment methods (e.g., Kenward-Roger) require theta_vcov to be calculated. These will then raise errors.

Value

List of class mmrm_control with the control parameters.

Examples

```
mmrm_control(
  optimizer_fun = stats::optim,
  optimizer_args = list(method = "L-BFGS-B")
)
```

---

mmrm_tidiers *Tidying Methods for* mmrm *Objects*

Description

[Stable]

These methods tidy the estimates from an mmrm object into a summary.

Usage

```
## S3 method for class 'mmrm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'mmrm'
glance(x, ...)

## S3 method for class 'mmrm'
augment(
  x,
  newdata = NULL,
  interval = c("none", "confidence", "prediction"),
  se_fit = (interval != "none"),
  type.residuals = c("response", "pearson", "normalized"),
  ...
)
```

Arguments

x	(mmrm) fitted model.
conf.int	(flag) if TRUE columns for the lower (conf.low) and upper bounds (conf.high) of coefficient estimates are included.
conf.level	(number) defines the range of the optional confidence internal.
...	only used by augment() to pass arguments to the predict.mmrm_tmb() method.
newdata	(data.frame or NULL) optional new data frame.
interval	(string) type of interval calculation.
se_fit	(flag) whether to return standard errors of fit.
type.residuals	(string) passed on to residuals.mmrm_tmb().

Functions

- tidy(mmrm): derives tidy tibble from an mmrm object.
- glance(mmrm): derives glance tibble from an mmrm object.
- augment(mmrm): derives augment tibble from an mmrm object.

See Also

mmrm_methods, mmrm_tmb_methods for additional methods.

Examples

```
fit <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
# Applying tidy method to return summary table of covariate estimates.
fit |> tidy()
fit |> tidy(conf.int = TRUE, conf.level = 0.9)
# Applying glance method to return summary table of goodness of fit statistics.
fit |> glance()
# Applying augment method to return merged `tibble` of model data, fitted and residuals.
fit |> augment()
fit |> augment(interval = "confidence")
fit |> augment(type.residuals = "pearson")
```

mmrm_tmb_methods *Methods for* mmrm_tmb *Objects*

Description

[Stable]

Usage

```
## S3 method for class 'mmrm_tmb'
coef(object, complete = TRUE, ...)

## S3 method for class 'mmrm_tmb'
fitted(object, ...)

## S3 method for class 'mmrm_tmb'
predict(
  object,
  newdata,
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  nsim = 1000L,
  conditional = FALSE,
  ...
)

## S3 method for class 'mmrm_tmb'
model.frame(
  formula,
  data,
  include = c("subject_var", "visit_var", "group_var", "response_var"),
  exclude = NULL,
  full,
  na.action = "na.omit",
  ...
)

## S3 method for class 'mmrm_tmb'
model.matrix(
  object,
  data,
  use_response = TRUE,
  contrasts.arg = attr(object$tmb_data$x_matrix, "contrasts"),
  ...
)
```

```
## S3 method for class 'mmrm_tmb'
terms(x, include = "response_var", ...)

## S3 method for class 'mmrm_tmb'
logLik(object, ...)

## S3 method for class 'mmrm_tmb'
formula(x, ...)

## S3 method for class 'mmrm_tmb'
vcov(object, complete = TRUE, ...)

## S3 method for class 'mmrm_tmb'
VarCorr(x, sigma = NA, ...)

## S3 method for class 'mmrm_tmb'
deviance(object, ...)

## S3 method for class 'mmrm_tmb'
AIC(object, corrected = FALSE, ..., k = 2)

## S3 method for class 'mmrm_tmb'
BIC(object, ...)

## S3 method for class 'mmrm_tmb'
print(x, ...)

## S3 method for class 'mmrm_tmb'
residuals(object, type = c("response", "pearson", "normalized"), ...)

## S3 method for class 'mmrm_tmb'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata,
  ...,
  method = c("conditional", "marginal")
)
```

Arguments

object	(mmrm_tmb) the fitted MMRM object.
complete	(flag) whether to include potential non-estimable coefficients.
...	mostly not used; Exception is `model.matrix()` passing `...` to the default method.

newdata	(data.frame) optional new data, otherwise data from object is used.
se.fit	(flag) indicator if standard errors are required.
interval	(string) type of interval calculation. Can be abbreviated.
level	(number) tolerance/confidence level.
nsim	(count) number of simulations to use.
conditional	(flag) indicator if the prediction is conditional on the observation or not.
formula	(mmrm_tmb) same as object.
data	(data.frame) object in which to construct the frame.
include	(character) names of variable types to include in the model frame creation. Must be NULL or one or more of c("subject_var", "visit_var", "group_var", "response_var").
exclude	(character) names of variable types to exclude after the model frame creation. Same choices as for include.
full	(flag) indicator whether to return full model frame (deprecated).
na.action	(string) na action.
use_response	(flag) whether to use the response for complete rows.
contrasts.arg	(list) contrasts to be used.
x	(mmrm_tmb) same as object.
sigma	cannot be used (this parameter does not exist in MMRM).
corrected	(flag) whether corrected AIC should be calculated.
k	(number) the penalty per parameter to be used; default k = 2 is the classical AIC.
type	(string) unscaled (response), pearson or normalized. Default is response, and this is the only type available for use with models with a spatial covariance structure.
seed	unused argument from stats::simulate().
method	(string) simulation method to use. If "conditional", simulated values are sampled given the estimated covariance matrix of object. If "marginal", the variance of the estimated covariance matrix is taken into account.

Details

`include` argument controls the variables the model frame will include upon creation. Possible options are "response_var", "subject_var", "visit_var" and "group_var", representing the response variable, subject variable, visit variable or group variable. By default all four variable types are included, which means that the original model frame will be used. If only a subset of variable types is requested, the model frame will be constructed freshly, and if there are more complete rows (without `NA`s) in the required variables than in the original model frame, those rows will be included as well.

The `exclude` argument can be used to exclude specific variable types after the model frame creation. By default no variable types are excluded. The `exclude` argument is useful when the original model frame should be used as basis (with same incomplete rows omitted), but some variable types should be removed afterwards.

`character` values in new data will always be factorized according to the data in the fit to avoid mismatched in levels or issues in `model.matrix`.

Value

Depends on the method, see Functions.

Functions

- `coef(mmrm_tmb)`: obtains the estimated coefficients.
- `fitted(mmrm_tmb)`: obtains the fitted values.
- `predict(mmrm_tmb)`: predict conditional means for new data; optionally with standard errors and confidence or prediction intervals. Returns a vector of predictions if `se.fit == FALSE` and `interval == "none"`; otherwise it returns a data.frame with multiple columns and one row per input data row.
- `model.frame(mmrm_tmb)`: obtains the model frame.
- `model.matrix(mmrm_tmb)`: obtains the model matrix.
- `terms(mmrm_tmb)`: obtains the terms object.
- `logLik(mmrm_tmb)`: obtains the attained log likelihood value. Includes as attributes the number of variance parameters `n_param`, number of estimated coefficients `n_coef`, degrees of freedom `df`, and number of subjects `nobs`. The `nobs` attribute is so named so that if this function's results are passed to `stats::BIC()`, the BIC value will be calculated correctly. Resulting value is of class `logLik`.
- `formula(mmrm_tmb)`: obtains the used formula.
- `vcov(mmrm_tmb)`: obtains the variance-covariance matrix estimate for the coefficients.
- `VarCorr(mmrm_tmb)`: obtains the variance-covariance matrix estimate for the residuals.
- `deviance(mmrm_tmb)`: obtains the deviance, which is defined here as twice the negative log likelihood, which can either be integrated over the coefficients for REML fits or the usual one for ML fits.
- `AIC(mmrm_tmb)`: obtains the Akaike Information Criterion, where the degrees of freedom are the number of variance parameters (`n_theta`). If `corrected`, then this is multiplied with `m / (m - n_theta - 1)` where `m` is the number of observations minus the number of coefficients, or `n_theta + 2` if it is smaller than that (Hurvich and Tsai 1989; Burnham and Anderson 1998).

- `BIC(mmrm_tmb)`: obtains the Bayesian Information Criterion, which is using the natural logarithm of the number of subjects for the penalty parameter k.
- `print(mmrm_tmb)`: prints the object.
- `residuals(mmrm_tmb)`: to obtain residuals - either unscaled ('response'), 'pearson' or 'normalized'.
- `simulate(mmrm_tmb)`: simulate responses from a fitted model according to the simulation `method`, returning a `data.frame` of dimension `[n, m]` where n is the number of rows in `newdata`, and m is the number `nsim` of simulated responses.

Note

If the response in the formula is a complicated expression, e.g. with multiple variables, and not all variables are provided in `newdata`, a warning is issued. In this case it is recommended to provide all variables in the `newdata` data set, and manually set the response variables to `NA` as needed for obtaining unconditional predictions e.g.

References

- Hurvich CM, Tsai C (1989). "Regression and time series model selection in small samples." *Biometrika*, **76**(2), 297–307. doi:10.2307/2336663.
- Burnham KP, Anderson DR (1998). "Practical use of the information-theoretic approach." In *Model selection and inference*, 75–117. Springer. doi:10.1007/9781475729177_3.
- Gałecki A, Burzykowski T (2013). "Linear mixed-effects model." In *Linear mixed-effects models using R*, 245–273. Springer.

See Also

`mmrm_methods`, `mmrm_tidiers` for additional methods.

Examples

```
formula <- FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID)
object <- fit_mmrm(formula, fev_data, weights = rep(1, nrow(fev_data)))
# Estimated coefficients:
coef(object)
# Fitted values:
fitted(object)
predict(object, newdata = fev_data)
# Model frame:
model.frame(object)
model.frame(object, include = "subject_var")
# Model matrix:
model.matrix(object)
# terms:
terms(object)
terms(object, include = "subject_var")
# Log likelihood given the estimated parameters:
logLik(object)
# Formula which was used:
```

```
formula(object)
# Variance-covariance matrix estimate for coefficients:
vcov(object)
# Variance-covariance matrix estimate for residuals:
VarCorr(object)
# REML criterion (twice the negative log likelihood):
deviance(object)
# AIC:
AIC(object)
AIC(object, corrected = TRUE)
# BIC:
BIC(object)
# residuals:
residuals(object, type = "response")
residuals(object, type = "pearson")
residuals(object, type = "normalized")
```

`print.cov_struct` *Print a Covariance Structure Object*

Description

Print a Covariance Structure Object

Usage

```
## S3 method for class 'cov_struct'
print(x, ...)
```

Arguments

`x`	(`cov_struct`) a covariance structure object.
`...`	Additional arguments unused.

Value

`x` invisibly.

refit_multiple_optimizers *Refitting MMRM with Multiple Optimizers*

Description

[Stable]

Usage

```
refit_multiple_optimizers(fit, ..., control = mmrm_control(...))
```

Arguments

fit	(mmrm_fit) original model fit from fit_single_optimizer().
...	Additional arguments passed to mmrm_control().
control	(mmrm_control) object.

Value

The best (in terms of log likelihood) fit which converged.

Note

For Windows, no parallel computations are currently implemented.

Examples

```
fit <- fit_single_optimizer(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data,
  weights = rep(1, nrow(fev_data)),
  optimizer = "nlminb"
)
best_fit <- refit_multiple_optimizers(fit)
```

stats_anova *Analysis of Variance for* mmrm *Fits*

Description

If supplied only one model fit, the function will calculate and return the significance of the model terms. If supplied more than one model fit, standard diagnostics will be returned for each model, optionally including likelihood ratio test (LRT) results for adjacent models.

Usage

```
## S3 method for class 'mmrm'
anova(object, ..., test = TRUE, refit = FALSE)
```

Arguments

object	(mmrm) an mmrm model fit.
...	(mmrm) optional mmrm model fits. If left empty, the significance of each term in object will be calculated.
test	(flag) indicating whether the output should include likelihood ratio test (LRT) results comparing the model fits to one another. Defaults to TRUE. Ignored if ... is empty.
refit	(flag) indicating whether the models should be refitted with the dataset consisting of their shared set of observations before performing diagnostics and testing. This is ignored if the models already share the same dataset. If refit = FALSE and the models have different underlying data sets, an error will be thrown. Defaults to FALSE. Ignored if ... is empty.

Details

When test = FALSE (or, when only one model is supplied), this function will process any mmrm fits, related or unrelated.

When supplying multiple models and test = TRUE, adjacent pairs of models are tested sequentially. In other words, the order of the supplied models matters. Furthermore, there are are multiple requirements for successful LRT. See the section "Requirements for LRT" below.

Value

A data frame with a row for each supplied model. If ... is empty, this will be the the returned value of h_anova_single_mmrm_model() with object supplied as its argument. Otherwise, the resulting data frame will have the following columns:

- `Model`: the sequence number of the model according to the order in which the models were supplied to this function.
- `refit`: logical, indicating whether or not the model was refitted. If the `refit` argument was `FALSE`, all values will be `FALSE`.
- `REML`: logical, indicating whether or not the model was fitted using restricted maximum likelihood (REML) estimation. If `FALSE`, the model was fitted using maximum likelihood (ML) estimation.
- `n_param`: the number of variance parameters in the model fit, obtained via `logLik.mmrm_tmb()`.
- `n_coef`: the number of estimated coefficients in the model fit, obtained via `logLik.mmrm_tmb()`.
- `df`: degrees of freedom of the model fit, obtained via `logLik.mmrm_tmb()`.
- `AIC`: Akaike's "An Information Criterion" of the model fit, obtained via `stats::AIC()`.
- `BIC`: the Bayesian Information Criterion of the model fit, obtained via `stats::BIC()`.
- `logLik`: the log likelihood of the model fit, obtained via `logLik.mmrm_tmb()`.
- `call`: the call that created the model fit, obtained via `component()` with `name = "call"`, which is passed to `deparse1()`. If the model was refitted (i.e., if its `refit` entry in this table is `TRUE`), this `call` will be different from the `call` component of the pre-refitted model fit.

The data frame will have these additional columns inserted before `call` if `test = TRUE`. Note that since each of these columns describe the results of a likelihood ratio test (LRT) between the previous row's and current row's model fits, the first element of each of these columns will be `NA`.

- `test`: character, indicating which two model fits were compared. Values are of the form `{Model - 1} vs {Model}`.
- `log_likelihood_ratio`: the logarithm of the likelihood ratio between the two models being compared.
- `p_value`: the p-value of the `log_likelihood_ratio`.

Requirements for LRT

1. Each supplied model fit must have more degrees of freedom than the preceding model fit.
2. If all supplied models were estimated using maximum likelihood (ML), the models must have nested covariates in order to undergo LRT. In other words, the set of covariates for each model must be a subset of the covariates of the next model. However, if any of the supplied models were estimated using restricted maximum likelihood (REML), all models must have the same covariates.
3. The covariance structure of each model must be either (a) the same as that of the next model or (b) a special case of that of the next model. See the section "Covariance structure nesting hierarchy" below.
4. All supplied model fits must either already use the same data or be refitted using `refit = TRUE`, which refits all models to the dataset of common observations between all models' respective data sets.

Covariance structure nesting hierarchy

Structured nests within unstructured:

Tautologically, all covariance structures are special cases of an unstructured covariance, and a model *with* a covariance structure can be considered "nested" within an model *without* a covariance structure (assuming that the covariates are also nested).

Homogeneous nests within analogous heterogeneous:

All homogeneous covariance structures are nested within their corresponding heterogeneous counterparts. For instance, the homogeneous Toeplitz covariance structure is nested within the heterogeneous Toeplitz covariance structure.

Other nested structures:

Some different covariance structure types are also nested:

- First-order auto-regressive (`ar1` / `ar1h`) is nested within:
  - ante-dependence (`ad` / `adh`)
  - Toeplitz (`toep` / `toeph`)
- Compound symmetry (`cs` / `csh`) is nested within Toeplitz (`toep` / `toeph`)

See Also

For details on the single model operation of this function, see `h_anova_single_mmrm_model()`. For details on the generic, see `stats::anova()`.

Examples

```
# Create a few model fits, only adding terms from one to the next.
# Notice also that each covariance structure is a special case of the one
# that follows.
fit_sex_ar1 <-
  mmrm(FEV1 ~ FEV1_BL + SEX + ARMCD + ar1(AVISIT | USUBJID),
    data = fev_data,
    reml = FALSE
  )
fit_sex_race_toeph <-
  mmrm(
    FEV1 ~ FEV1_BL + SEX + RACE + ARMCD + toeph(AVISIT | USUBJID),
    data = fev_data,
    reml = FALSE
  )
fit_interaction_us <-
  mmrm(
    FEV1 ~ FEV1_BL + SEX * RACE + ARMCD + us(AVISIT | USUBJID),
    data = fev_data,
    reml = FALSE
  )

# Single model fit, showing significance of model terms:
anova(fit_interaction_us)

# Multiple model fits, with diagnostics for each fit and likelihood ratio
```

```
# testing (LRT) for each adjacent pair. LRT is possible because when the fits
# are in this order, their covariates and covariance structures are nested.
anova(fit_sex_ar1, fit_sex_race_toeph, fit_interaction_us)

# We can only change the order if we forego LRT using test = FALSE.
anova(fit_sex_race_toeph, fit_interaction_us, fit_sex_ar1, test = FALSE)

# Create a subset of fev_data set with the 4th visit removed.
fev_subset <- droplevels(fev_data[fev_data$VISITN < 4, ])

# Recreate fit_sex_race_toeph but this time based off fev_subset:
fit_sex_race_toeph_sub <-
  mmrm(
    FEV1 ~ FEV1_BL + SEX + RACE + ARMCD + toeph(AVISIT | USUBJID),
    data = fev_subset,
    reml = FALSE
  )

# If a model was created with a different data set, refit = TRUE is needed.
anova(fit_sex_ar1, fit_sex_race_toeph_sub, fit_interaction_us, refit = TRUE)
```

`std_start` *Standard Starting Value*

Description

Obtain standard start values.

Usage

```
std_start(cov_type, n_visits, n_groups, ...)
```

Arguments

`cov_type`	(`string`) name of the covariance structure.
`n_visits`	(`int`) number of visits.
`n_groups`	(`int`) number of groups.
`...`	not used.

Details

`std_start` will try to provide variance parameter from identity matrix. However, for `ar1` and `ar1h` the corresponding values are not ideal because the ρ is usually a positive number thus using 0 as starting value can lead to incorrect optimization result, and we use 0.5 as the initial value of ρ.

Value

A numeric vector of starting values.

Index