Package 'emaxnls'

June 30, 2026

Title Nonlinear Least Squares Estimation for Emax Regression Models

Version 0.1.1

Description Provides estimation and covariate selection tools for Emax regression models using nonlinear least squares methods. Supported optimization algorithms are Gauss-Newton, Levenberg-Marquardt, and the port library for bounded optimization. The package also provides tools to assist in simulation work using Emax regression.

License MIT + file LICENSE

URL https://github.com/djnavarro/emaxnls, https://emaxnls.djnavarro.net/

BugReports https://github.com/djnavarro/emaxnls/issues

Depends R (>= 3.5)

Imports Deriv, minpack.lm, mvtnorm, rlang, stats, tibble, utils

Suggests spelling, testthat (>= 3.0.0)

Config/Needs/website djnavarro/waeponwifestre

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Language en-US

NeedsCompilation no

Author Danielle Navarro [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7648-6578>)

Maintainer Danielle Navarro <djnavarro@protonmail.com>

Repository CRAN

Date/Publication 2026-06-30 20:10:02 UTC

Contents

`AIC` *Akaike information criterion / Bayesian information criterion*

Description

Akaike information criterion / Bayesian information criterion

Usage

```
## S3 method for class 'emaxnls'
AIC(object, ..., k = 2)

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

Arguments

`object`	An `emaxnls` object
`...`	Optionally, more fitted model objects
`k`	Penalty per parameter in the AIC

Value

If just one object is provided, a numeric value with the corresponding AIC (or BIC). If multiple objects are provided, a data.frame with rows corresponding to the objects and columns representing the number of parameters in the model (df) and the AIC or BIC.

Examples

```
mod_0 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
mod_1 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

# calculate AIC for individual models
AIC(mod_0)
AIC(mod_1)

# calculate AIC for a sequence of models
AIC(mod_0, mod_1)

# calculate BIC for individual models
BIC(mod_0)
BIC(mod_1)

# calculate BIC for a sequence of models
BIC(mod_0, mod_1)
```

---

anova.emaxnls *Analysis of variance for Emax regression models*

Description

Analysis of variance for Emax regression models

Usage

```
## S3 method for class 'emaxnls'
anova(object, ...)
```

Arguments

object	An emaxnls object
...	Additional fitted model objects

Value

Analysis of variance tables for a sequence of `emaxnls` models

Examples

```
mod_0 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
mod_1 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

anova(mod_0, mod_1)
```

---

coef.emaxnls *Coefficients for an Emax regression*

Description

Coefficients for an Emax regression

Usage

```
## S3 method for class 'emaxnls'
coef(object, back_transform = FALSE, ...)
```

Arguments

`object`	An `emaxnls` object
`back_transform`	Should log-scaled parameters (logEC50, logHill) be back-transformed to original scale?
`...`	Ignored

Value

A vector of coefficients

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

# coefficients on the estimation scale
coef(mod)

# coefficients with log-scale parameters back-transformed
coef(mod, back_transform = TRUE)
```

`confint.emaxnls` *Confidence intervals for Emax regression model parameters*

Description

Confidence intervals for Emax regression model parameters

Usage

```
## S3 method for class 'emaxnls'
confint(object, parm = NULL, level = 0.95, back_transform = FALSE, ...)
```

Arguments

`object`	An `emaxnls` object
`parm`	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If `parm = NULL`, all parameters are considered.
`level`	The confidence level required
`back_transform`	Should log-scaled parameters (logEC50, logHill) be back-transformed to original scale?
`...`	Ignored

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labeled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

# 95% confidence interval on the estimation scale
confint(mod)

# 90% confidence interval on the estimation scale
confint(mod, level = 0.9)

# 95% confidence interval with log-scale parameters back-transformed
confint(mod, back_transform = TRUE)
```

`deviance.emaxnls` *Model deviance for an Emax regression*

Description

Model deviance for an Emax regression

Usage

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

Arguments

`object`	An `emaxnls` object
`...`	Ignored

Value

Numeric

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
deviance(mod)
```

df.residual.emaxnls *Residual degrees of freedom for an Emax regression model*

Description

Residual degrees of freedom for an Emax regression model

Usage

```
## S3 method for class 'emaxnls'
df.residual(object, ...)
```

Arguments

object	An emaxnls object
...	Ignored

Value

Numeric

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
df.residual(mod)
```

emax_converged *Check Emax regression model for convergence status*

Description

Check Emax regression model for convergence status

Usage

```
emax_converged(mod)
```

Arguments

mod	An emaxnls object

Details

This is a convenience function that takes an Emax regression object as input. It returns TRUE if the optimization routine converged during model fitting, and FALSE if it did not.

Value

Logical value

---

emax_df *Sample simulated data for Emax exposure-response models with covariates.*

Description

Sample simulated data for Emax exposure-response models with covariates.

Usage

```
emax_df
```

Format

A data frame with columns:

id Identifier column

dose Nominal dose, units not specified

exp_1 Exposure value, units and metric not specified

exp_2 Exposure value, units and metric not specified, but different from exp_1

rsp_1 Continuous response value (units not specified)

rsp_2 Binary response value (group labels not specified)

cnt_a Continuous valued covariate

cnt_b Continuous valued covariate

cnt_c Continuous valued covariate

bin_d Binary valued covariate

bin_e Binary valued covariate

cat_f Categorical covariate

Details

This simulated dataset is entirely synthetic. It is a generic data set that can be used to illustrate Emax modeling. It contains variables corresponding to dose and exposure, and includes both a continuous response variable and a binary response variable. Three continuous valued covariates are included, along with two binary covariates.

You can find the data generating code in the package source code, under R/data.R

Examples

```
emax_df
```

`emax_fun` *Construct Emax prediction function from model object*

Description

Construct Emax prediction function from model object

Usage

```
emax_fun(mod)
```

Arguments

`mod`	An `emaxnls` object

Value

A function `f` with arguments `data` and `params`. The `data` argument defaults to the data used to estimate the model, and the `params` argument defaults to the estimated parameter values. Both can be customized, as long as `data` contains columns corresponding to each of the variables used by the model, and `params` is a named numeric vector of the appropriate length. The names for `params` must exactly match the names of the vector returned by `coef(mod)`.

The return value for `f` is a numeric vector of model predictions for each row in `data`, evaluated at parameters `params`.

See Also

`emax_nls()`

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

# the emax simulation function can only be extracted if the
# optimization routine converged
if (emax_converged(mod)) {

  par <- coef(mod)

  # customizable emax function with the same structural
  # model and same covariate model, defaulting to the
```

```
  # same data and parameters as the original model, but
  # allowing user to pass their own data and parameters
  mod_fn <- emax_fun(mod)

  # apply the function to a few rows of the original data
  out_1 <- mod_fn(
    data = emax_df[120:125, ],
    param = par
  )
  print(out_1)

  # adjust the parameters
  new_par <- par
  new_par["E0_Intercept"] <- 0

  # simulate the model with the adjusted parameters
  out_2 <- mod_fn(
    data = emax_df[120:125, ],
    param = new_par
  )
  print(out_2)

}
```

`emax_nls` *Estimate parameters for an Emax regression model*

Description

Estimate parameters for an Emax regression model

Usage

```
emax_nls(structural_model, covariate_model, data, init = NULL, opts = NULL)
```

Arguments

`structural_model`	A two-sided formula of the form response ~ exposure
`covariate_model`	A list of two-sided formulas, each of specifying a covariate model for a structural parameter
`data`	A data frame that includes all relevant variables
`init`	Initial values and bounds for parameters. See `emax_nls_init()`
`opts`	Model fitting and optimization options. See `emax_nls_options()`

Details

The `emax_nls()` function is the workhorse function for estimating an Emax regression model. Pass a two-sided formula to the `structural_model` argument to specify the exposure variable and the response variable (e.g., `response ~ exposure`), and pass a list of formulas to the `covariate_model` argument to specify covariates of interest. At a minimum the covariate model requires specification of the covariate model for the E0 parameter, the Emax parameter, and the logEC50 parameter. For example, a formula like `E0 ~ age + group` would indicate that `age` and `group` should both be included as covariates on the baseline response E0. When no covariates are to be added, use a formula like `Emax ~ 1`.

The `emax_nls()` function can support sigmoidal emax models as well as hyperbolic models. To build a sigmoidal model (where the Hill parameter) is estimated from the data, the `covariate_model` argument must also include a formula for the `logHill` parameter. For instance, if the covariate model includes `logHill ~ 1`, the model will estimate the value of the Hill parameter (with no covariates on it) from the data set.

At present, `emax_nls()` does not support binary response variables, nor is it possible to specify interaction terms in the covariate model.

When estimating model parameters, the `init` argument can be used to specify the starting values for the optimization. If unspecified, the `emax_nls_init()` function is used to automatically guess sensible starting values. Please see the documentation of that function for additional details on manually specifying the initial values.

The `emax_nls()` function currently supports three optimization methods: the Gauss-Newton algorithm, the Levenberg-Marquardt algorithm, and the 'nl2sol' algorithm from the Port library. For more information on how to customize the optimization procedure, please see the documentation for `emax_nls_options()`.

Value

An object of class `emaxnls`

See Also

`emax_nls_options()`, `emax_nls_init()`

Examples

```
emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
```

`emax_nls_init` *Construct an initial guess for the Emax model parameters*

Description

Construct an initial guess for the Emax model parameters

Usage

```
emax_nls_init(structural_model, covariate_model, data)
```

Arguments

`structural_model`	A two-sided formula of the form response ~ exposure
`covariate_model`	A list of two-sided formulas, each of specifying a covariate model for a structural parameter
`data`	A data frame

Details

The `emax_nls()` function requires that the user specify the initial values for the model parameters. Specifically, it expects to be supplied with a data frame with columns named `parameter`, `covariate`, and `start`. If a bounded optimization method is used (e.g. if the "port" method is used), the data frame also needs to have columns named `lower` and `upper`. The data frame should contain one row per parameter. In most cases the user does not need to define this manually, because `emax_nls_init()` can use heuristics to make a sensible guess about what to use as starting values. By default this is what `emax_nls()` relies upon, automatically calling `emax_nls_init()` using the appropriate values for the `structural_model`, the `covariate_model`, and the `data`.

Value

A data frame

See Also

`emax_nls()`, `emax_nls_options()`

Examples

```
# use a heuristic to construct sensible start values, and plausible
# upper and lower bounds within which the estimate is expected to fall
emax_nls_init(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
```

```
# compare to the values estimated:
coef(emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
))
```

`emax_nls_options` *Settings used to estimate Emax model*

Description

Settings used to estimate Emax model

Usage

```
emax_nls_options(
  optim_method = "gauss",
  optim_control = NULL,
  quiet = FALSE,
  weights = NULL,
  na.action = options("na.action")
)
```

Arguments

`optim_method`	Character string specifying the algorithm used to solve the nonlinear least squares optimization problem. Supported options are "gauss" (the default), "port", and "levenberg". See details.
`optim_control`	A list of arguments used to control the behavior of the optimization algorithm. Allowed values differ depending on which algorithm is used
`quiet`	When `quiet=TRUE`, messages are suppressed
`weights`	Numeric vector providing the weights for observations. When specified, weighted least squares is used
`na.action`	How should missing values in the data be handled?

Details

At present there are three supported values for `optim_method`:

- "gauss": Estimate parameters using the Gauss-Newton algorithm. This is equivalent to the using "default" option in `nls()`
- "port": Estimate parameters using bounded optimization with the "nl2sol" algorithm from from the the Port library. Equivalent to "port" in `nls()`

- "levenberg": Estimate parameters using the Levenberg-Marquardt algorithm. This is equivalent to using `nlsLM()` from the "minpack.lm" package.

Note that the Golub-Pereyra algorithm for partially linear least-squares (i.e. the "plinear" option in `nls()`) is not currently supported for Emax regression. Informal testing suggests it does not perform well for these models, and rarely converges.

The `optim_control` argument mirrors the corresponding control arguments for the respective optimization methods:

- For "gauss" and "port": the list should match the output of `stats::nls.control()`
- For "levenberg": the list should match the output of `minpack.lm::nls.lm.control()`

If `optim_control = NULL`, the default settings are used for the relevant function.

Value

A list of settings

See Also

`emax_nls()`, `emax_nls_init()`

Examples

```
# default options
emax_nls_options()

# switch to levenberg-marquardt
if (require("minpack.lm", quietly = TRUE)) emax_nls_options(optim_method = "levenberg")
```

`emax_scm` *Stepwise covariate modeling for Emax regression*

Description

Stepwise covariate modeling for Emax regression

Usage

```
emax_scm_forward(mod, candidates, threshold = 0.01, seed = NULL)

emax_scm_backward(mod, candidates, threshold = 0.001, seed = NULL)

emax_scm_history(mod)
```

Arguments

mod	An emaxnls object
candidates	A list of candidate covariates
threshold	Threshold for addition or removal
seed	Seed for the RNG state

Details

The emaxnls package supports stepwise covariate modeling via forward addition and backward elimination. The emax_scm_forward() function supports forward addition, the emax_scm_backward() function supports backward elimination, and the syntax is designed to allow forward-backward procedures by piping a base model to emax_scm_forward() and then to emax_scm_backward(). In both cases, the function takes an emaxnls regression object as the first argument, as well as a list of candidate covariates to be considered for addition (in the forward addition case) or deletion (backward elimination). The input must be a named list, with the names corresponding to the relevant structural parameter, and the values should be character vector specifying covariates of interest. See the examples for an illustration of how this argument should be specified.

As present, these functions only support stepwise regression using p-values as the criterion for addition or deletion. The threshold argument corresponds to the threshold p-value to be used. In future, other methods (e.g., selection on the basis of AIC values) may be supported.

The seed argument is used to control the RNG state for stochastic components of the stepwise procedure. However, please note that the seed argument is currently experimental, and may be removed in future releases.

A key feature of the stepwise covariate modeling functions is that they keep track of every tested model, and store information about this history internally within the emaxnls object that gets returned. Use the emax_scm_history() function to extract this history.

Value

An object of class emaxnls

See Also

emax_nls()

Examples

```
base_model <- emax_nls(rsp_1 ~ exp_1, list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1), emax_df)

covariate_list <- list(
  E0 = c("cnt_a", "cnt_b", "cnt_c", "bin_d", "bin_e"),
  Emax = c("cnt_a", "cnt_b", "cnt_c", "bin_d", "bin_e")
)

# add covariates to the base model using forward addition
forward_model <- emax_scm_forward(
  mod = base_model,
  candidates = covariate_list,
```

```
  threshold = .01
)
forward_model

# remove covariates from the forward model using backward deletion
final_model <- emax_scm_backward(
  mod = forward_model,
  candidates = covariate_list,
  threshold = .001
)
final_model

# show the history of all models tested during the forward addition
# step and the backward deletion step
emax_scm_history(final_model)
```

`emax_update` *Add or remove a covariate term from an Emax regression*

Description

Add or remove a covariate term from an Emax regression

Usage

```
emax_add_term(mod, formula)

emax_remove_term(mod, formula)
```

Arguments

`mod`	An `emaxnls` object
`formula`	A formula such as E0 ~ AGE

Details

The `emax_add_term()` and `emax_remove_term()` functions take an existing Emax regression object, and allow the user to add or remove a specific term to the model. It is not expected that users will need these functions very often, but they provide the basis for the stepwise covariate modeling procedures that are very commonly used when building Emax regressions.

Value

An object of class `emaxnls`

See Also

`emax_nls()`, emax_scm

Examples

```
mod_0 <- emax_nls(rsp_1 ~ exp_1, list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1), emax_df)
mod_1 <- emax_nls(rsp_1 ~ exp_1, list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1), emax_df)

if (emax_converged(mod_0)) emax_add_term(mod_0, E0 ~ cnt_a)

if (emax_converged(mod_1)) emax_remove_term(mod_1, E0 ~ cnt_a)
```

`fitted.emaxnls` *Fitted values for an Emax regression*

Description

Fitted values for an Emax regression

Usage

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

Arguments

`object`	An `emaxnls` object
`...`	Ignored

Value

Numeric vector of fitted values

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
fit <- fitted(mod)
fit[1:20]
```

logLik.emaxnls *Log-likelihood for an Emax regression model*

Description

Log-likelihood for an Emax regression model

Usage

```
## S3 method for class 'emaxnls'
logLik(object, REML = FALSE, ...)
```

Arguments

object	An emaxnls object
REML	For nls objects only REML = FALSE is supported
...	Ignored

Value

Returns an object of class logLik. This is a number with at least one attribute, "df" (degrees of freedom), giving the number of (estimated) parameters in the model.

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
logLik(mod)
```

nobs.emaxnls *Number of observations for an Emax regression model*

Description

Number of observations for an Emax regression model

Usage

```
## S3 method for class 'emaxnls'
nobs(object, ...)
```

Arguments

object	An emaxnls object
...	Ignored

Value

Numeric

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
nobs(mod)
```

---

predict.emaxnls *Predicting from Emax regression models*

Description

Predicting from Emax regression models

Usage

```
## S3 method for class 'emaxnls'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  interval = "none",
  level = 0.95,
  ...
)
```

Arguments

object	An emaxnls object
newdata	A named list or data frame in which to look for variables with which to predict. If newdata is missing the fitted values at the original data points are returned.
se.fit	A switch indicating if standard errors are required.
interval	A character string indicating if prediction intervals or a confidence interval on the mean responses are to be calculated. Can be "none", "confidence", or "prediction"

level	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated.
...	Ignored

Value

The return value differs slightly depending on inputs. When se.fit = FALSE, it produces a vector or matrix of predictions with column names fit, lwr and upr if the interval argument is set. When se.fit = TRUE, it returns a list with the following components:

- fit: vector or matrix as above
- se.fit: standard error of the predicted means
- residual.scale: residual standard deviation
- df: residual degrees of freedom

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

# return a vector of predictions
pred <- predict(mod)
pred[1:20]

# return a matrix with confidence intervals
predict(mod, interval = "confidence", se.fit = FALSE)
```

---

print.emaxnls *Print an Emax regression model object*

Description

Print an Emax regression model object

Usage

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

Arguments

x	An emaxnls object
...	Ignored

Value

Invisibly returns the original object

`residuals.emaxnls` *Residuals for an Emax regression*

Description

Residuals for an Emax regression

Usage

```
## S3 method for class 'emaxnls'
residuals(object, ...)
```

Arguments

`object`	An `emaxnls` object
`...`	Ignored

Value

Numeric vector of residuals

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
res <- residuals(mod)
res[1:20]
```

`sigma.emaxnls` *Residual standard deviation for Emax regression models*

Description

Residual standard deviation for Emax regression models

Usage

```
## S3 method for class 'emaxnls'
sigma(object, ...)
```

Arguments

object	An emaxnls object
...	Ignored

Value

Numeric

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
sigma(mod)
```

---

simulate.emaxnls *Simulate responses from Emax regression model*

Description

Simulate responses from Emax regression model

Usage

```
## S3 method for class 'emaxnls'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	An emaxnls object
nsim	Number of replicates
seed	Used to set RNG seed
...	Ignored

The simulate() method for an emaxnls object simulates new values for the response, using new parameter values that sampled using the variance-covariance matrix associated with the parameter estimates. It uses mvtnorm::rmvnorm() to generate multivariate normal variates.

Value

A data frame or tibble

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
if (requireNamespace("mvtnorm", quietly = TRUE)) simulate(mod)
```

`summary.emaxnls` *Summary of an Emax regression model*

Description

Summary of an Emax regression model

Usage

```
## S3 method for class 'emaxnls'
summary(object, conf_level = 0.95, back_transform = FALSE, ...)
```

Arguments

`object`	An `emaxnls` object
`conf_level`	Confidence level for interval estimates
`back_transform`	Should log-scaled parameters (logEC50, logHill) be back-transformed to original scale?
`...`	Ignored

Value

A data frame or tibble containing a table of parameter estimates and other statistical summaries. Please note that the `summary()` method is experimental (moreso than other methods), and the return value may be modified in future releases as the package matures.

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)

# standard summary
summary(mod)

# summary with adjusted confidence level
summary(mod, conf_level = 0.99)
```

```
# summary with log-scale parameters transformed to original scale
summary(mod, back_transform = TRUE)
```

`vcov.emaxnls` *Variance-covariance matrix for an Emax regression*

Description

Variance-covariance matrix for an Emax regression

Usage

```
## S3 method for class 'emaxnls'
vcov(object, ...)
```

Arguments

`object`	An `emaxnls` object
`...`	Ignored

Value

A matrix

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
vcov(mod)
```

Index