

Package ‘emaxnls’

September 17, 2026

Title Nonlinear Least Squares Estimation for Emax Regression Models

Version 0.2.0

Description Provides estimation and covariate selection tools for Emax regression models using nonlinear least squares methods. Supported optimisation algorithms are Gauss-Newton, Levenberg-Marquardt, and the port library for bounded optimisation. 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, utils

Suggests erplots, spelling, testthat (>= 3.0.0), tibble

Config/Needs/website djnavarro/waeponwifestre, rmarkdown, ggplot2

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Language en-GB

Config/roxygen2/version 8.1.0

NeedsCompilation no

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

Maintainer Danielle Navarro <djnavarro@protonmail.com>

Repository CRAN

Date/Publication 2026-09-16 23:10:03 UTC

Contents

AIC.emaxlogistic	2
anova.emaxlogistic	4
coef.emaxnls	5
confint.emaxnls	6
deviance.emaxlogistic	8
df.residual.emaxlogistic	9
emax_converged	10
emax_df	11
emax_fun	12
emax_logistic	14
emax_logistic_init	15
emax_logistic_options	16
emax_nls	18
emax_nls_init	19
emax_nls_options	20
emax_scm	22
emax_update	25
fitted.emaxlogistic	26
logLik.emaxlogistic	27
nobs.emaxnls	28
predict.emaxlogistic	29
print.emaxlogistic	31
residuals.emaxlogistic	32
sigma.emaxnls	33
simulate.emaxlogistic	34
summary.emaxlogistic	35
vcov.emaxnls	38
Index	40

AIC.emaxlogistic	<i>Akaike information criterion / Bayesian information criterion</i>
------------------	--

Description

Computes AIC or BIC for one or more fitted Emax models. Lower values indicate a better-fitting model; values are only meaningful in comparison to other models fitted to the same response variable and dataset.

Usage

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

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

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

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

Arguments

object	An emaxnls or emaxlogistic object
...	Optionally, more fitted model objects
k	Penalty per parameter in the AIC. The default is $k = 2$. Ignored for BIC(), which always uses $\log(n)$.

Details

AIC applies a penalty of $2 * k$ to minus twice the log-likelihood, where k is the number of estimated parameters. BIC applies $\log(n) * k$, making it more conservative than AIC in large samples. When multiple models are passed, any non-converging models are dropped with a warning.

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,
  opts = emax_nls_options(max_time = 10)
)
mod_1 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

# 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)

# emaxlogistic models
mod_b0 <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
mod_b1 <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
AIC(mod_b0, mod_b1)
BIC(mod_b0, mod_b1)
```

anova.emaxlogistic	<i>Analysis of variance for Emax regression models</i>
--------------------	--

Description

Compares a sequence of nested Emax models. At least two model objects must be provided; all must be of the same class.

Usage

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

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

Arguments

object	An emaxnls or emaxlogistic object
...	Additional fitted model objects of the same class

Details

For emaxnls objects, calls `stats::anova()` on the underlying nls objects to produce an ANOVA table for the sequence of models. For emaxlogistic objects, computes a likelihood ratio chi-squared test comparing nested models; the test statistic is the difference in deviances and the reference distribution is chi-squared with degrees of freedom equal to the difference in the number of parameters. The nesting assumption is not checked; results are only interpretable when each successive model genuinely adds parameters to the previous one.

Value

For emaxnls objects, an analysis of variance table for a sequence of models. For emaxlogistic objects, a data frame with columns Df, Deviance, Df_diff, LRT, and Pr(>Chi).

Examples

```
mod_0 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
mod_1 <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
anova(mod_0, mod_1)

# emaxlogistic: likelihood ratio test
mod_b0 <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
mod_b1 <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
anova(mod_b0, mod_b1)
```

coef.emaxnls

Coefficients for an Emax regression

Description

Returns the named vector of fitted parameter values. Parameters involving a log transformation (logEC50, logHill) are returned on the log scale by default, which is the scale on which they are estimated.

Usage

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

Arguments

<code>object</code>	An <code>emaxnls</code> or <code>emaxlogistic</code> object
<code>back_transform</code>	Should log-scaled parameters (<code>logEC50</code> , <code>logHill</code>) be back-transformed to original scale? The default is <code>back_transform = FALSE</code> .
<code>...</code>	Ignored

Details

Setting `back_transform = TRUE` exponentiates `logEC50` and `logHill` and drops the `log` prefix from their names, expressing them on the concentration scale rather than the log-concentration scale on which they are estimated. `confint()` and `vcov()` are not affected by this argument and always return results on the log-concentration scale. The `summary()` method also accepts `back_transform = TRUE` and applies the same transformation to its coefficient table, including the confidence interval columns.

Value

A named numeric vector of parameter estimates

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

# coefficients on the estimation scale
coef(mod_c)

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

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
coef(mod_b)
```

Description

Computes profile likelihood confidence intervals for the model parameters. Profile likelihood intervals are generally preferred over Wald intervals in nonlinear settings because they do not assume the likelihood surface is quadratic near the estimates.

Usage

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

Arguments

<code>object</code>	An <code>emaxnls</code> or <code>emaxlogistic</code> object
<code>parm</code>	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If <code>parm = NULL</code> , all parameters are considered.
<code>level</code>	The confidence level required. The default is <code>level = 0.95</code> .
<code>back_transform</code>	Should log-scaled parameters (<code>logEC50</code> , <code>logHill</code>) be back-transformed to original scale? The default is <code>back_transform = FALSE</code> .
<code>simultaneous</code>	If <code>TRUE</code> , return simultaneous (joint) Wald confidence intervals rather than the default profile likelihood intervals. Defaults to <code>FALSE</code> .
<code>...</code>	Ignored

Details

By default, and when `simultaneous = FALSE`, this calls `stats::confint.nls()` for `emaxnls` objects. For `emaxlogistic` objects, the same profiling approach is applied to the final NLS fit from the IRLS algorithm at convergence. If profile likelihood computation fails (which can occur for sigmoidal models), a warning is issued and Wald intervals are returned instead.

When `simultaneous = TRUE`, a single critical value is derived from the joint multivariate normal distribution of the standardised parameter estimates (via `mvtnorm::qmvnorm()`). The resulting intervals have simultaneous coverage at `level` across all parameters and will be wider than the individual (pointwise) intervals. This matches the intervals produced by `summary(object, simultaneous = TRUE)`.

Setting `back_transform = TRUE` exponentiates the confidence limits for `logEC50` and `logHill`, expressing them on the concentration scale rather than the log-concentration scale on which they are estimated, and drops the `log` prefix from their row names.

Value

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

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

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

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

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

# simultaneous (joint) confidence intervals
confint(mod_c, simultaneous = TRUE)

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
confint(mod_b)
```

deviance.emaxlogistic *Model deviance for an Emax regression model*

Description

Returns a scalar measure of the overall lack of fit. The two model classes use different definitions of deviance that reflect their different likelihoods. Both measures decrease as the fit improves and are used internally by `anova()` for model comparison.

Usage

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

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

Arguments

object	An emaxnls or emaxlogistic object
...	Ignored

Value

A numeric scalar. For emaxnls objects, returns the residual sum of squares. For emaxlogistic objects, returns the binomial deviance ($-2 * \text{logLik}$).

Examples

```
# emaxnls deviance (residual sum of squares)
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
deviance(mod_c)

# emaxlogistic deviance (binomial deviance)
mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
deviance(mod_b)
```

df.residual.emaxlogistic

Residual degrees of freedom for an Emax regression model

Description

Returns the residual degrees of freedom, equal to the number of observations minus the number of estimated parameters.

Usage

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

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

Arguments

object	An emaxnls or emaxlogistic object
...	Ignored

Details

For emaxnls objects, the value is obtained directly from the underlying nls fit. For emaxlogistic objects, it is computed as `nobs(object) - length(coef(object))`.

Value

A numeric scalar

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
df.residual(mod_c)

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
df.residual(mod_b)
```

emax_converged	<i>Check Emax regression model for convergence status</i>
----------------	---

Description

Returns TRUE if the model converged during fitting and FALSE otherwise. The reason for convergence or non-convergence is attached as the names attribute of the return value, so it prints alongside the logical result.

Usage

```
emax_converged(mod)
```

Arguments

mod	An emaxnls object
-----	-------------------

Value

A named logical scalar. The value is TRUE when the model converged and FALSE otherwise. The names attribute holds a short description of the outcome:

- "converged": the optimiser reached a solution successfully.
- "maximum time exceeded": the `max_time` limit set in `emax_nls_options()` / `emax_logistic_options()` was hit before the optimiser finished.
- "maximum iterations exceeded": the optimiser ran out of iterations. This applies to the Gauss-Newton algorithm (when `stats::nls()` reports "number of iterations exceeded maximum") and to the Levenberg-Marquardt algorithm (when `minpack.lm::nlsLM()` reports that the iteration count has reached `maxiter`). The iteration budget can be increased via the `optim_control` argument of `emax_nls_options()`.
- Raw optimiser message: all other failures return the error message from the underlying optimiser directly. Common examples include a singular gradient matrix at the initial parameter estimates, the Gauss-Newton step factor collapsing below `minFactor`, and Port-algorithm convergence codes such as false convergence (code 8) or singular convergence (code 7). These messages are algorithm-specific and are passed through unchanged because no single label covers them accurately.

See Also

`emax_nls()`, `emax_nls_options()`

emax_df	<i>Sample simulated data for Emax exposure-response models with covariates.</i>
---------	---

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 modelling. 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

Extracts a customizable prediction function from a fitted Emax model, allowing predictions to be evaluated at arbitrary data and parameter values.

Usage

```
emax_fun(mod, ...)

## S3 method for class 'emaxlogistic'
emax_fun(mod, ...)

## S3 method for class 'emaxnls'
emax_fun(mod, ...)
```

Arguments

<code>mod</code>	An emaxnls or emaxlogistic object
<code>...</code>	Ignored

Details

The extracted function accepts data and param arguments. Both default to the values used when fitting the model. When supplying custom values, data must contain all variables used by the model, and param must be a named numeric vector whose names exactly match those returned by `coef(mod)`.

Scale of predictions. For `emaxnls` objects the returned function produces predictions on the response scale (the same scale as the outcome variable). For `emaxlogistic` objects the structural Emax model is parameterized on the logit scale — $\text{logit}(p) = E0 + Emax * x / (x + EC50)$ — but `emax_fun()` applies the inverse-logit transformation before returning, so predictions are on the probability scale. This is consistent with the default behaviour of `fitted()` and `predict()` for `emaxlogistic` objects. If you need the linear predictor (logit scale) directly, use `fitted(object, type = "link")` or `predict(object, type = "link")`.

Value

A function with arguments `param` and `data` that evaluates the Emax model at the supplied (or default) parameter values and data. For `emaxnls` objects the return values are on the response scale; for `emaxlogistic` objects they are predicted probabilities in (0, 1).

See Also

`emax_nls()`, `emax_logistic()`

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

if (emax_converged(mod_c)) {

  par <- coef(mod_c)
  mod_fn <- emax_fun(mod_c)

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

  # adjust the parameters and re-evaluate
  new_par <- par
  new_par["E0_Intercept"] <- 0
  mod_fn(data = emax_df[120:125, ], param = new_par)
}

# for emaxlogistic, the returned function gives probabilities
mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
```

```

covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
data = emax_df,
opts = emax_logistic_options(max_time = 10)
)

if (emax_converged(mod_b)) {
  mod_fn_b <- emax_fun(mod_b)
  mod_fn_b(data = emax_df[120:125, ])
}

```

emax_logistic

Estimate parameters for a logistic Emax regression model

Description

Fits a logistic Emax regression model for a binary response variable using iterative reweighted least squares (IRLS). For continuous outcomes, use [emax_nls\(\)](#) instead.

Usage

```

emax_logistic(
  structural_model,
  covariate_model = NULL,
  data,
  init = NULL,
  opts = NULL
)

```

Arguments

<code>structural_model</code>	A two-sided formula of the form <code>response ~ exposure</code>
<code>covariate_model</code>	A list of two-sided formulas, each specifying a covariate model for a structural parameter. When <code>NULL</code> (the default), an intercept-only hyperbolic Emax model is fitted — equivalent to <code>list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1)</code> .
<code>data</code>	A data frame that includes all relevant variables
<code>init</code>	Initial values and bounds for parameters. See emax_logistic_init()
<code>opts</code>	Model fitting and optimisation options. See emax_logistic_options()

Details

The structural Emax model is placed on the log-odds (logit) scale:

$\text{logit}(p) = E0 + Emax * x / (x + EC50)$ (hyperbolic)

$\text{logit}(p) = E0 + Emax * x^h / (x^h + EC50^h)$ (sigmoidal)

Estimation uses iterative reweighted least squares (IRLS). At each outer iteration a weighted NLS problem is solved using working weights and a working response derived from the current parameter estimates. This is equivalent to Fisher scoring and produces maximum likelihood estimates at convergence.

The interface mirrors the `emax_nls()` function for continuous response models: the `structural_model` and `covariate_model` arguments have the same specification, including support for sigmoidal models via a `logHill` term. The response variable in `structural_model` must be a binary (0/1) numeric vector.

Value

An object of class `emaxlogistic` (which also inherits from `emaxnls`)

See Also

`emax_logistic_options()`, `emax_logistic_init()`, `emax_nls()`

Examples

```
# simplest call: hyperbolic Emax with no covariates
emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)

# with a covariate on the baseline parameter
emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
```

<code>emax_logistic_init</code>	<i>Construct an initial guess for logistic Emax model parameters</i>
---------------------------------	--

Description

Constructs a data frame of starting values and parameter bounds for the logistic Emax model, using the same heuristic approach as `emax_nls_init()` applied to the empirical logit scale rather than the raw response.

Usage

```
emax_logistic_init(structural_model, covariate_model = NULL, data)
```

Arguments

`structural_model` A two-sided formula of the form `response ~ exposure`

`covariate_model` A list of two-sided formulas, each specifying a covariate model for a structural parameter. When NULL (the default), an intercept-only hyperbolic Emax model is assumed — equivalent to `list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1)`.

`data` A data frame

Value

A data frame with columns `parameter`, `covariate`, `start`, `lower`, and `upper`

See Also

[emax_logistic\(\)](#), [emax_logistic_options\(\)](#)

Examples

```
# intercept-only hyperbolic Emax (default covariate_model)
emax_logistic_init(
  structural_model = rsp_2 ~ exp_1,
  data = emax_df
)

# with a covariate on E0
emax_logistic_init(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df
)
```

`emax_logistic_options` *Settings used to estimate a logistic Emax model*

Description

Constructs a settings object controlling the NLS optimiser and IRLS convergence for [emax_logistic\(\)](#). Pass the result to the `opts` argument of [emax_logistic\(\)](#).

Usage

```
emax_logistic_options(
  optim_method = "gauss",
  optim_control = NULL,
  quiet = FALSE,
  na.action = getOption("na.action"),
```

```

    max_iter = 25,
    tol = 1e-06,
    max_time = Inf
  )

```

Arguments

<code>optim_method</code>	Character string specifying the algorithm used for the weighted NLS step within each IRLS iteration. Supported options are "gauss" (default), "port", and "levenberg". See emax_nls_options() for details on each.
<code>optim_control</code>	A list of arguments controlling the NLS optimiser. The default is <code>optim_control = NULL</code> , which uses the default settings for the relevant optimisation function.
<code>quiet</code>	When TRUE, convergence warnings are suppressed.
<code>na.action</code>	How should missing values in the data be handled? The default is <code>na.action = getOption("na.action")</code> .
<code>max_iter</code>	Maximum number of IRLS outer iterations (default 25).
<code>tol</code>	Convergence tolerance: IRLS stops when the change in binomial deviance between successive iterations falls below <code>tol</code> (default 1e-6).
<code>max_time</code>	Maximum elapsed time in seconds allowed for the entire model fit (including all IRLS iterations). If the optimiser has not converged within this time, it is terminated and the model is treated as non-converged. Defaults to <code>Inf</code> (no time limit).

Value

A list of settings

See Also

[emax_logistic\(\)](#), [emax_logistic_init\(\)](#)

Examples

```

# default options
emax_logistic_options()

# increase maximum IRLS iterations
emax_logistic_options(max_iter = 50)

```

 emax_nls

Estimate parameters for an Emax regression model

Description

Fits an Emax regression model for a continuous response variable using nonlinear least squares. For binary outcomes, use [emax_logistic\(\)](#) instead.

Usage

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

Arguments

<code>structural_model</code>	A two-sided formula of the form <code>response ~ exposure</code>
<code>covariate_model</code>	A list of two-sided formulas, each specifying a covariate model for a structural parameter. When <code>NULL</code> (the default), an intercept-only hyperbolic Emax model is fitted — equivalent to <code>list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1)</code> .
<code>data</code>	A data frame that includes all relevant variables
<code>init</code>	Initial values and bounds for parameters. See emax_nls_init()
<code>opts</code>	Model fitting and optimisation options. See emax_nls_options()

Details

Pass a two-sided formula to `structural_model` to specify the response and exposure variables (e.g., `response ~ exposure`), and a list of formulas to `covariate_model` to specify covariates. At a minimum the covariate model requires formulas for `E0`, `Emax`, and `logEC50`. A formula like `E0 ~ age + group` includes `age` and `group` as covariates on the baseline response; use `Emax ~ 1` when no covariates are to be added for a parameter.

To fit a sigmoidal Emax model (estimating the Hill parameter), include a formula for `logHill` in `covariate_model`, e.g. `logHill ~ 1`. Without this term a hyperbolic model is fitted. Interaction terms in the covariate model are not currently supported.

Starting values are constructed automatically via [emax_nls_init\(\)](#) unless the `init` argument is supplied manually. Three optimisation algorithms are available; see [emax_nls_options\(\)](#) for details.

Value

An object of class `emaxnls`

See Also

[emax_nls_options\(\)](#), [emax_nls_init\(\)](#)

Examples

```
# simplest call: hyperbolic Emax with no covariates
emax_nls(
  structural_model = rsp_1 ~ exp_1,
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

# with a covariate on the baseline parameter
emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
```

emax_nls_init	<i>Construct an initial guess for the Emax model parameters</i>
---------------	---

Description

Constructs a data frame of starting values and parameter bounds for the Emax NLS optimisation, using heuristics derived from the data.

Usage

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

Arguments

structural_model	A two-sided formula of the form response ~ exposure
covariate_model	A list of two-sided formulas, each specifying a covariate model for a structural parameter. When NULL (the default), an intercept-only hyperbolic Emax model is assumed — equivalent to <code>list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1)</code> .
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 optimisation 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

```
# intercept-only hyperbolic Emax (default covariate_model)
emax_nls_init(
  structural_model = rsp_1 ~ exp_1,
  data = emax_df
)

# with a covariate on E0
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,
  opts = emax_nls_options(max_time = 10)
))
```

emax_nls_options

Settings used to estimate Emax model

Description

Constructs a settings object controlling the optimisation algorithm and other aspects of model fitting for `emax_nls()`. Pass the result to the `opts` argument of `emax_nls()`.

Usage

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

Arguments

<code>optim_method</code>	Character string specifying the algorithm used to solve the nonlinear least squares optimisation problem. Supported options are "gauss" (the default), "port", and "levenberg". See details.
<code>optim_control</code>	A list of arguments used to control the behaviour of the optimisation algorithm. Allowed values differ depending on which algorithm is used. The default is <code>optim_control = NULL</code> , which uses the default settings for the relevant optimisation function; see details.
<code>quiet</code>	When <code>quiet=TRUE</code> , messages are suppressed
<code>weights</code>	Numeric vector providing the weights for observations. When specified, weighted least squares is used. The default is <code>weights = NULL</code> (unweighted least squares).
<code>na.action</code>	How should missing values in the data be handled? The default is <code>na.action = getOption("na.action")</code> .
<code>max_time</code>	Maximum elapsed time in seconds allowed for the model fit. If the optimiser has not converged within this time, it is terminated and the model is treated as non-converged (the same outcome as any other convergence failure). Defaults to <code>Inf</code> (no time limit).

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 `stats::nls()`
- "port": Estimate parameters using bounded optimisation with the "nl2sol" algorithm from from the the Port library. Equivalent to "port" in `stats::nls()`
- "levenberg": Estimate parameters using the Levenberg-Marquardt algorithm. This is equivalent to using `minpack.lm::nlsLM()` from the "minpack.lm" package.

Note that the Golub-Pereyra algorithm for partially linear least-squares (i.e. the "plinear" option in `stats::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 optimisation 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 modelling for Emax regression*

Description

Performs stepwise covariate modelling by forward addition (`emax_scm_forward()`), backward elimination (`emax_scm_backward()`), or both in sequence. Use `emax_scm_history()` to retrieve the history of all models tested during the procedure.

Usage

```
emax_scm_forward(
  mod,
  candidates,
  threshold = 0.01,
  criterion = "p-value",
  seed = NULL
)

emax_scm_backward(
  mod,
  candidates,
  threshold = 0.001,
  criterion = "p-value",
  seed = NULL
)

emax_scm_history(mod)
```

Arguments

<code>mod</code>	An <code>emaxnls</code> object
<code>candidates</code>	A list of candidate covariates
<code>threshold</code>	Threshold for addition or removal. Used only when <code>criterion = "p-value"</code> (the default); ignored otherwise.
<code>criterion</code>	Model selection criterion. One of <code>"p-value"</code> (default), <code>"aic"</code> , or <code>"bic"</code> .
<code>seed</code>	Seed for the RNG state

Details

The `candidates` argument must be a named list whose names correspond to structural parameters (e.g. `E0`, `Emax`) and whose values are character vectors of covariate names to consider. See the examples for an illustration.

Three model selection criteria are available via the `criterion` argument:

- `"p-value"` (default): a term is added if its ANOVA p-value falls below `threshold` (forward) or removed if its p-value exceeds `threshold` (backward). When multiple candidates satisfy the threshold, the one with the most extreme p-value is chosen.
- `"aic"`: a term is added (forward) or removed (backward) if doing so strictly decreases AIC. When multiple candidates improve AIC, the one yielding the lowest AIC is chosen.
- `"bic"`: same as `"aic"` but using BIC as the criterion.

When `criterion` is `"aic"` or `"bic"`, the `threshold` argument has no effect and is ignored.

The history returned by `emax_scm_history()` always records AIC and BIC for every model tested (columns `model_aic` and `model_bic`), regardless of which criterion was used for selection. The `criterion` column records which criterion drove each forward or backward step.

The `seed` argument controls the RNG state for any stochastic components of the procedure. It is currently experimental and may be removed in future releases.

Every model tested during the procedure is stored internally in the returned object. Use `emax_scm_history()` to extract this record.

Value

An object of class `emaxnls`

See Also

[emax_nls\(\)](#)

Examples

```
base_model <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
```

```

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 (p-value criterion)
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, including which criterion was used
emax_scm_history(final_model)

# AIC-based forward addition
forward_aic <- emax_scm_forward(
  mod = base_model,
  candidates = covariate_list,
  criterion = "aic"
)

# BIC-based backward elimination
final_bic <- emax_scm_backward(
  mod = forward_aic,
  candidates = covariate_list,
  criterion = "bic"
)

emax_scm_history(final_bic)

# example using binary outcomes
base_model_logistic <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
forward_model_logistic <- emax_scm_forward(
  mod = base_model_logistic,
  candidates = covariate_list,
  threshold = .01
)
final_model_logistic <- emax_scm_backward(
  mod = forward_model_logistic,

```

```

      candidates = covariate_list,
      threshold = .001
    )

    final_model_logistic

    emax_scm_history(final_model_logistic)

```

 emax_update

Add or remove a covariate term from an Emax regression

Description

Add or remove a single covariate term from an existing Emax regression model, returning a new fitted model object.

Usage

```

emax_add_term(mod, formula)

emax_remove_term(mod, formula)

```

Arguments

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

Details

These functions are not typically called directly; they underpin the stepwise covariate modelling procedures that are very commonly used when building Emax regressions.

Value

An object of class emaxnls

See Also

[emax_nls\(\)](#), [emax_scm](#)

Examples

```

opts <- emax_nls_options(max_time = 10)
mod_0 <- emax_nls(rsp_1 ~ exp_1, list(E0 ~ 1, Emax ~ 1, logEC50 ~ 1), emax_df, opts = opts)
mod_1 <- emax_nls(rsp_1 ~ exp_1, list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1), emax_df, opts = opts)

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.emaxlogistic *Fitted values for an Emax regression model*

Description

Returns the model predictions at the original data points. For emaxlogistic objects, the type argument controls whether fitted probabilities or the linear predictor on the logit scale are returned.

Usage

```
## S3 method for class 'emaxlogistic'
fitted(object, type = c("response", "link"), ...)

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

Arguments

object	An emaxnls or emaxlogistic object
type	For emaxlogistic objects: "response" (default) returns fitted probabilities; "link" returns the linear predictor on the logit scale. Ignored for emaxnls objects.
...	Ignored

Details

For emaxnls objects, these are the predicted values from the Emax curve evaluated at each observation's exposure and covariate values.

Value

A numeric vector of fitted values, with length equal to the number of observations in the original data

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
fitted(mod_c)[1:20]

mod_b <- emax_logistic(
```

```

structural_model = rsp_2 ~ exp_1,
covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
data = emax_df,
opts = emax_logistic_options(max_time = 10)
)
fitted(mod_b)[1:20]
fitted(mod_b, type = "link")[1:20]

```

logLik.emaxlogistic	<i>Log-likelihood for an Emax regression model</i>
---------------------	--

Description

Evaluates the log-likelihood of a fitted Emax model at the maximum likelihood estimates. The returned object is compatible with AIC(), BIC(), and likelihood ratio tests via anova().

Usage

```

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

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

```

Arguments

object	An emaxnls or emaxlogistic object
REML	For emaxnls objects only REML = FALSE is supported. Ignored for emaxlogistic objects.
...	Ignored

Details

For emaxnls objects, the log-likelihood is computed under the assumption of normally distributed errors, as returned by stats::logLik.nls(). For emaxlogistic objects, it is the binomial log-likelihood evaluated at the fitted probabilities. The logLik object carries df (number of parameters) and nobs attributes.

Value

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

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
logLik(mod_c)

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
logLik(mod_b)
```

nobs.emaxnls

Number of observations for an Emax regression model

Description

Returns the number of observations used when fitting the model.

Usage

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

Arguments

object	An emaxnls or emaxlogistic object
...	Ignored

Details

This reflects the actual number of rows passed to the fitting algorithm after any missing-value handling specified via the `na.action` option in `emax_nls_options()` or `emax_logistic_options()`. The value is used internally by `BIC()` and `df.residual()`.

Value

A numeric scalar

Examples

```

mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
nobs(mod_c)

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
nobs(mod_b)

```

predict.emaxlogistic *Predicting from Emax regression models*

Description

Generates predictions from a fitted Emax model, either at the original data points or at new covariate and exposure values supplied via newdata. Standard errors and confidence or prediction intervals can be requested.

Usage

```

## S3 method for class 'emaxlogistic'
predict(
  object,
  newdata = NULL,
  type = c("response", "link"),
  se.fit = FALSE,
  interval = "none",
  level = 0.95,
  ...
)

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

```

Arguments

<code>object</code>	An <code>emaxnls</code> or <code>emaxlogistic</code> object
<code>newdata</code>	A named list or data frame in which to look for variables with which to predict. If <code>newdata</code> is missing the fitted values at the original data points are returned.
<code>type</code>	For <code>emaxlogistic</code> objects: "response" (default) returns predicted probabilities; "link" returns the linear predictor on the logit scale. Ignored for <code>emaxnls</code> objects.
<code>se.fit</code>	A switch indicating if standard errors are required. The default is <code>se.fit = FALSE</code> .
<code>interval</code>	A character string indicating if prediction intervals or a confidence interval on the mean responses are to be calculated. Can be "none" (the default), "confidence", or "prediction".
<code>level</code>	A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. The default is <code>level = 0.95</code> .
<code>...</code>	Ignored

Details

For `emaxlogistic` objects, when `interval` is set, the bounds are first computed on the link scale and then passed through the inverse logit transformation, ensuring they remain in the unit interval on the probability scale.

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_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

# return a vector of predictions
predict(mod_c)[1:20]

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

```
# emaxlogistic predicted probabilities
mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
predict(mod_b)[1:20]
predict(mod_b, type = "link")[1:20]
```

print.emaxlogistic	<i>Print an Emax regression model object</i>
--------------------	--

Description

The `print()` method for `emaxnls` and `emaxlogistic` objects provides a concise model overview: the structural and covariate formulas, key fit statistics, and a coefficient table showing estimates and confidence intervals. Hypothesis tests are deliberately omitted from the printed output; use `summary()` for inferential results.

Usage

```
## S3 method for class 'emaxlogistic'
print(x, conf_level = 0.95, ...)

## S3 method for class 'emaxnls'
print(x, conf_level = 0.95, ...)
```

Arguments

<code>x</code>	An <code>emaxnls</code> or <code>emaxlogistic</code> object
<code>conf_level</code>	Confidence level for the coefficient intervals shown in the printed table. Defaults to 0.95.
<code>...</code>	Ignored

Value

Invisibly returns the original object

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
```

```

)
print(mod_c)

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
print(mod_b)

```

residuals.emaxlogistic

Residuals for an Emax regression model

Description

For emaxnls objects, returns raw residuals on the response scale. For emaxlogistic objects, Pearson or deviance residuals are available via the type argument.

Usage

```

## S3 method for class 'emaxlogistic'
residuals(object, type = c("pearson", "deviance"), ...)

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

```

Arguments

object	An emaxnls or emaxlogistic object
type	For emaxlogistic objects: the type of residuals to return. "pearson" (default) returns Pearson residuals; "deviance" returns deviance residuals. Ignored for emaxnls objects.
...	Ignored

Details

Pearson residuals are the raw residuals divided by $\sqrt{\mu * (1 - \mu)}$, the estimated standard deviation of a Bernoulli observation, giving a standardised measure of discrepancy. Deviance residuals are the signed square root of each observation's contribution to the total binomial deviance; their sum of squares equals the model deviance returned by deviance().

Value

A numeric vector of residuals

Examples

```

mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
residuals(mod_c)[1:20]

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
residuals(mod_b)[1:20]
residuals(mod_b, type = "deviance")[1:20]

```

sigma.emaxnls

*Residual standard deviation for an Emax regression model***Description**

Returns the estimated residual standard deviation from the NLS fit. This method is only available for emaxnls objects; there is no analogous quantity for emaxlogistic models.

Usage

```

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

```

Arguments

object	An emaxnls object
...	Ignored

Details

Under the assumption of normally distributed errors, sigma estimates the standard deviation of the error term in the Emax model. It is computed as the square root of the residual sum of squares divided by the residual degrees of freedom.

Value

A numeric scalar

Examples

```
mod <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
sigma(mod)
```

simulate.emaxlogistic *Simulate responses from an Emax regression model*

Description

Generates simulated response datasets from a fitted Emax model, propagating uncertainty in the parameter estimates. This is useful for constructing simulation-based confidence bands, for predictive checks, or for bootstrapping downstream analyses.

Usage

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

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

Arguments

object	An emaxnls or emaxlogistic object
nsim	Number of replicates. The default is nsim = 1.
seed	Used to set RNG seed. The default is seed = NULL, which does not set the seed.
...	Ignored

Details

The `simulate()` method samples new parameter values from the multivariate normal distribution implied by the estimated covariance matrix, then simulates responses at those parameter values using `mvtnorm::rmvnorm()`. For `emaxlogistic` objects, predicted probabilities are computed from each parameter draw and binary outcomes are drawn from `Bernoulli(p)` for each observation.

Value

A data frame with `nsim` columns named `sim_1`, `sim_2`, etc.

Examples

```

mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)
simulate(mod_c)

mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
simulate(mod_b)

```

summary.emaxlogistic *Summary of an Emax regression model*

Description

Returns a tidy coefficient table for a fitted emaxnls or emaxlogistic model, combining parameter estimates, standard errors, test statistics, p-values, and confidence intervals.

Usage

```

## S3 method for class 'emaxlogistic'
summary(
  object,
  conf_level = 0.95,
  back_transform = FALSE,
  p_adjust = "none",
  simultaneous = FALSE,
  suppress_nonsensical = TRUE,
  ...
)

## S3 method for class 'emaxnls'
summary(
  object,
  conf_level = 0.95,
  back_transform = FALSE,
  p_adjust = "none",
  simultaneous = FALSE,
  suppress_nonsensical = TRUE,
  ...
)

```

Arguments

<code>object</code>	An <code>emaxnls</code> or <code>emaxlogistic</code> object
<code>conf_level</code>	Confidence level for interval estimates. Defaults to 0.95.
<code>back_transform</code>	Should <code>logEC50</code> and <code>logHill</code> parameters be back-transformed to the concentration scale? If <code>TRUE</code> , these parameters are exponentiated and renamed to <code>EC50</code> and <code>Hill</code> respectively, and their confidence intervals are transformed accordingly. Standard errors and test statistics on the back-transformed scale are not available and are set to <code>NA</code> .
<code>p_adjust</code>	Method for adjusting p-values for multiple comparisons, passed to <code>stats::p.adjust()</code> . Defaults to "none". Parameters with suppressed p-values (see <code>suppress_nonsensical</code>) are excluded from the adjustment set. This affects only the <code>p_value</code> column and is independent of <code>simultaneous</code> ; see the "Multiplicity: p-value adjustment versus simultaneous intervals" section below.
<code>simultaneous</code>	If <code>TRUE</code> , compute simultaneous (joint) confidence intervals using the multivariate normal distribution via <code>mvtnorm::qmvnorm()</code> . This gives wider intervals that provide joint coverage at <code>conf_level</code> across all parameters simultaneously. Defaults to <code>FALSE</code> . This affects only the confidence-interval columns and is independent of <code>p_adjust</code> ; see the "Multiplicity: p-value adjustment versus simultaneous intervals" section below.
<code>suppress_nonsensical</code>	If <code>TRUE</code> (the default), suppress the test statistic and p-value for <code>logEC50_Intercept</code> . The <code>logEC50</code> intercept is estimated on the log-concentration scale, and testing $H_0: \log EC50 = 0$ is equivalent to testing whether <code>EC50</code> equals 1 on the concentration scale — a threshold with no general pharmacometric meaning. The confidence interval for <code>logEC50</code> is always reported regardless of this setting. Pass <code>suppress_nonsensical = FALSE</code> to restore the raw test results.
<code>...</code>	Ignored

Details

Which tests are reported by default:

Most parameters have a meaningful point null at zero:

- `Emax_Intercept`: tests whether any exposure-response relationship exists.
- `logHill_Intercept`: tests whether `logHill` = 0, i.e., whether the Hill parameter equals 1 on the concentration scale, which would mean the sigmoidal model reduces to a hyperbolic one.
- `E0_Intercept`: tests whether the baseline response is zero. This is informative when the outcome is expressed as change from baseline, a common convention in pharmacometrics.
- Covariate beta terms: test whether a given covariate has any effect on the corresponding structural parameter.

The one exception is `logEC50_Intercept`. The model is parameterized in terms of `logEC50` (on the log-concentration scale), not `EC50` directly, so the null $H_0: \log EC50 = 0$ corresponds to testing `EC50` = 1 on the concentration scale — a value with no intrinsic pharmacometric meaning. By default, the test statistic and p-value for `logEC50_Intercept` are suppressed (set to `NA`), while the confidence interval for `logEC50` is retained. To work with the `EC50` on the concentration scale, use `back_transform = TRUE`.

Simultaneous intervals:

When `simultaneous = TRUE`, a single critical value is derived from the joint multivariate normal distribution of the standardised parameter estimates. The resulting intervals have simultaneous coverage at `conf_level` and will be wider than the individual (pointwise) intervals.

Multiplicity: p-value adjustment versus simultaneous intervals:

A model with several parameters raises a multiple-comparisons problem: the more quantities you inspect, the more likely at least one spurious result appears by chance. `summary()` offers two, deliberately separate, tools for this, and it is worth being clear about how they differ because they are easy to conflate.

- `p_adjust` acts on the **hypothesis tests**. It takes the marginal (per-parameter) p-values and feeds them through `stats::p.adjust()`, which applies a sequential rule such as Holm or a Bonferroni scaling. These rules look only at the *set of p-values*; they do not use the estimated correlations between the parameters. Only the `p_value` column changes. The estimates, standard errors, test statistics, and confidence intervals are untouched.
- `simultaneous` acts on the **interval estimates**. It replaces the per-parameter critical value with a single, larger critical value taken from the joint multivariate normal distribution of the estimates (via `mvtnorm::qmvnorm()`), which *does* use the correlation structure from `vcov()`. Only the `ci_lower` and `ci_upper` columns change. The p-values and test statistics are untouched.

The two arguments are fully independent: you may set either, both, or neither, and each does exactly the one thing described above. Neither argument modifies the other's output.

Why the adjusted p-values and the intervals may disagree:

Because the two corrections use different machinery, they will not, in general, agree on which parameters are "significant". A parameter can have a Holm-adjusted p-value below $1 - \text{conf_level}$ while its simultaneous confidence interval still contains zero, or the reverse. This is not a bug. The familiar duality — "the 95% interval excludes zero if and only if the two-sided test rejects at the 5% level" — holds only for a *single*, unadjusted Wald comparison. It breaks as soon as a multiplicity correction enters, for two reasons:

- the corrections answer different questions (a step-down rule on the p-values versus a joint critical region for the intervals), and
- they use different information (`p.adjust()` ignores the parameter correlations that the simultaneous interval is built from).

A further, smaller source of discrepancy: unless the profile-likelihood computation falls back to Wald intervals, the default (pointwise) intervals are profile-likelihood based, whereas the reported test statistics and p-values are Wald quantities, so even *without* any adjustment the two need not correspond exactly in nonlinear models.

Which one to use:

Pick the tool that matches the claim you want to make, and interpret its output on its own terms rather than cross-checking one against the other:

- To report **interval estimates** that are jointly valid across all parameters, use `simultaneous = TRUE`.
- To control the family-wise (or false-discovery) error rate of a set of **hypothesis tests**, choose a `p_adjust` method.

Setting both is legitimate, but the adjusted p-values and the simultaneous intervals are then two separate summaries of multiplicity, not two views of the same one, and should be read as such.

Value

A tibble with one row per model parameter and columns for the estimate, standard error, test statistic, p-value, and confidence interval bounds. The column for the test statistic is named `t_statistic` for `emaxnls` models and `z_statistic` for `emaxlogistic` models. The return format is experimental and may change in future releases.

Examples

```
mod_c <- emax_nls(
  structural_model = rsp_1 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_nls_options(max_time = 10)
)

# standard summary (logEC50_Intercept p-value suppressed by default)
summary(mod_c)

# show all tests, including the logEC50 intercept
summary(mod_c, suppress_nonsensical = FALSE)

# Bonferroni-adjusted p-values
summary(mod_c, p_adjust = "bonferroni")

# simultaneous confidence intervals
summary(mod_c, simultaneous = TRUE)

# adjusted confidence level
summary(mod_c, conf_level = 0.99)

# back-transform logEC50 and logHill to concentration scale
summary(mod_c, back_transform = TRUE)

# logistic emax equivalent
mod_b <- emax_logistic(
  structural_model = rsp_2 ~ exp_1,
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),
  data = emax_df,
  opts = emax_logistic_options(max_time = 10)
)
summary(mod_b)
```

vcov.emaxnls

*Variance-covariance matrix for an Emax regression***Description**

Returns the estimated variance-covariance matrix of the model parameters. The square roots of the diagonal entries are the parameter standard errors.

Usage

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

Arguments

object	An emaxnls or emaxlogistic object
...	Ignored

Details

For emaxnls objects, the matrix is derived from the Hessian of the NLS objective at the parameter estimates (via `stats::vcov.nls()`). For emaxlogistic objects, it is derived from the Jacobian of the IRLS algorithm at convergence, which provides the correct asymptotic covariance matrix under binomial sampling.

Value

A square numeric matrix with rows and columns named by the model parameters

Examples

```
mod_c <- emax_nls(  
  structural_model = rsp_1 ~ exp_1,  
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),  
  data = emax_df,  
  opts = emax_nls_options(max_time = 10)  
)  
vcov(mod_c)  
  
mod_b <- emax_logistic(  
  structural_model = rsp_2 ~ exp_1,  
  covariate_model = list(E0 ~ cnt_a, Emax ~ 1, logEC50 ~ 1),  
  data = emax_df,  
  opts = emax_logistic_options(max_time = 10)  
)  
vcov(mod_b)
```

Index

* datasets

 emax_df, [11](#)

AIC (AIC.emaxlogistic), [2](#)
AIC.emaxlogistic, [2](#)
anova (anova.emaxlogistic), [4](#)
anova.emaxlogistic, [4](#)

BIC.emaxlogistic (AIC.emaxlogistic), [2](#)
BIC.emaxnls (AIC.emaxlogistic), [2](#)

coef.emaxnls, [5](#)
confint.emaxnls, [6](#)

deviance (deviance.emaxlogistic), [8](#)
deviance.emaxlogistic, [8](#)
df.residual (df.residual.emaxlogistic),
 [9](#)
df.residual.emaxlogistic, [9](#)

emax_add_term (emax_update), [25](#)
emax_converged, [10](#)
emax_df, [11](#)
emax_fun, [12](#)
emax_logistic, [14](#)
emax_logistic(), [13](#), [16–18](#)
emax_logistic_init, [15](#)
emax_logistic_init(), [14](#), [15](#), [17](#)
emax_logistic_options, [16](#)
emax_logistic_options(), [11](#), [14–16](#)
emax_nls, [18](#)
emax_nls(), [11](#), [13–15](#), [20](#), [22](#), [23](#), [25](#)
emax_nls_init, [19](#)
emax_nls_init(), [15](#), [18–20](#), [22](#)
emax_nls_options, [20](#)
emax_nls_options(), [11](#), [17–20](#)
emax_remove_term (emax_update), [25](#)
emax_scm, [22](#), [25](#)
emax_scm_backward (emax_scm), [22](#)
emax_scm_backward(), [22](#)
emax_scm_forward (emax_scm), [22](#)
emax_scm_forward(), [22](#)
emax_scm_history (emax_scm), [22](#)
emax_scm_history(), [22](#), [23](#)
emax_update, [25](#)

fitted (fitted.emaxlogistic), [26](#)
fitted(), [13](#)
fitted.emaxlogistic, [26](#)

logLik (logLik.emaxlogistic), [27](#)
logLik.emaxlogistic, [27](#)

minpack.lm::nls.lm.control(), [22](#)
minpack.lm::nlsLM(), [11](#), [21](#)
mvtnorm::qmvnorm(), [7](#), [36](#), [37](#)

nobs.emaxnls, [28](#)

predict (predict.emaxlogistic), [29](#)
predict(), [13](#)
predict.emaxlogistic, [29](#)
print (print.emaxlogistic), [31](#)
print.emaxlogistic, [31](#)

residuals (residuals.emaxlogistic), [32](#)
residuals.emaxlogistic, [32](#)

sigma.emaxnls, [33](#)
simulate (simulate.emaxlogistic), [34](#)
simulate.emaxlogistic, [34](#)
stats::nls(), [11](#), [21](#)
stats::nls.control(), [21](#)
stats::p.adjust(), [36](#), [37](#)
summary (summary.emaxlogistic), [35](#)
summary.emaxlogistic, [35](#)

vcov.emaxnls, [38](#)