

Package ‘erglm’

September 29, 2026

Title Exposure-Response Tools for GLM-Based Models

Version 0.2.0

Description Provides estimation tools for exposure-response models based on `glm()`: model fitting and prediction, stepwise covariate modelling, and simulation-based visual predictive checks. Tested and supported for binomial, Poisson, Gaussian, and gamma families. For a model-agnostic mini-language to visualise exposure-response models (including those fitted with 'erglm'), see the companion package 'erplots'.

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

Imports mvtnorm, stats

Suggests erplots, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0),
withr

URL <https://github.com/djnavarro/erglm>, <https://erglm.djnavarro.net/>

BugReports <https://github.com/djnavarro/erglm/issues>

Depends R (>= 4.1.0)

LazyData true

Config/testthat/edition 3

Config/Needs/website djnavarro/waeponwifestre, rmarkdown

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 2026-09-29 11:40:02 UTC

Contents

erglm_data	2
erglm_fun	3
erglm_link	4
erglm_model	5
erglm_predict	6
erglm_scm	7
erglm_term	10
simulate.erglm_model	11

Index	13
--------------	-----------

erglm_data	<i>Sample simulated data for exposure-response models with covariates</i>
------------	---

Description

A synthetic dataset bundled with the package and used throughout its documentation and examples, with response columns illustrating each of erglm's supported glm() families.

Usage

erglm_data

Format

A data frame with columns:

id Identifier

sex Sex

age Age

weight Weight

dose Nominal dose, units not specified

treatment Treatment

aucss AUCss

cmaxss Cmax,ss

ae1 Binary response 1 value (for binomial models)

ae2 Binary response 2 value (for binomial models)

ae_count Count response (for poisson models)

biomarker_change Continuous response, can be negative (for gaussian models)

ae_duration Continuous, strictly positive, right-skewed response (for gamma models)

Details

This simulated dataset is entirely synthetic. See the package source for the data-generating code.

Examples

```
head(erglm_data)
```

erglm_fun

Prediction function for an exposure-response model

Description

Takes a fitted glm object as input and returns a function that evaluates the underlying structural model at user-specified parameters or data (e.g., for VPCs or other counterfactual simulation scenarios).

Usage

```
erglm_fun(object)
```

Arguments

object An erglm model, as returned by [erglm_model\(\)](#)

Details

Uses `stats::family(object)$linkinv`, so this works for any `glm()` family, not just binomial/logistic models; tested for binomial, poisson, gaussian, and gamma families. Named `erglm_fun()` for consistency with the companion `emaxnls` package's [emaxnls::emax_fun\(\)](#), which serves the same purpose for `emaxnls/emaxlogistic` models. The returned function checks that `param` is numeric and has one entry per column of the model matrix implied by data, erroring informatively rather than failing with a cryptic "non-conformable arguments" error from matrix multiplication.

Value

A function with arguments `param`, `data`, and `type`.

- The `param` argument should be a vector of coefficients; defaults to `coef(object)` (the fitted coefficients) if not supplied.
- The `data` argument should be a data frame or tibble; defaults to `object$data` (the data the model was fitted to) if not supplied.
- The `type` argument should be a string indicating the type of prediction to generate (defaults to "response")

Examples

```
mod1 <- erglm_model(ae2 ~ aucss + sex, erglm_data, family = binomial())
mod1_fun <- erglm_fun(mod1)

# no arguments: reproduces the fitted model's own predictions
p1 <- mod1_fun()
p2 <- unname(predict(mod1, type = "response")) # same result

# user modifies the data set
erglm_data2 <- erglm_data[1:20, ]
p3 <- mod1_fun(data = erglm_data2)
p4 <- unname(predict(mod1, newdata = erglm_data2, type = "response")) # same result

# user modifies the parameters
par2 <- coef(mod1)
int1 <- par2["(Intercept)"]
par2["(Intercept)"] <- 0
p5 <- mod1_fun(param = par2)
```

erglm_link

Link and inverse-link functions for a fitted model

Description

Every `glm()` family already carries its link and inverse-link functions (`stats::family(mod)$linkfun` / `stats::family(mod)$linkinv`), but many users don't realise these are available for the taking. `erglm_link()` and `erglm_invlink()` are thin, discoverable wrappers around them: `erglm_link()` maps the response scale to the linear predictor scale, and `erglm_invlink()` maps the linear predictor scale back to the response scale.

Usage

```
erglm_link(mod)

erglm_invlink(mod)
```

Arguments

`mod` A fitted model, typically an `erglm_model`/`glm` object.

Value

A function of one numeric-vector argument.

Examples

```
mod <- erglm_model(ae1 ~ aucss + sex, erglm_data, family = binomial())
erglm_link(mod)(0.5)
erglm_invlink(mod)(0)
erglm_link(mod)(erglm_invlink(mod)(-2:2))
```

erglm_model	<i>Fit an exposure-response model based on glm()</i>
-------------	--

Description

A thin wrapper around `stats::glm()` that fits the model and tags the result with an extra `erglm_model` class, so downstream `erglm` functions (and the optional `erplots` interoperability layer) can recognise it.

Usage

```
erglm_model(formula, data, family = stats::gaussian(), ...)
```

Arguments

formula	Model formula
data	Data set
family	The error distribution and link function to use, as for <code>stats::glm()</code> . Defaults to <code>stats::gaussian()</code> , matching <code>stats::glm()</code> 's own default. Tested and officially supported for <code>binomial()</code> , <code>poisson()</code> , <code>gaussian()</code> , and <code>Gamma()</code> ; other <code>glm()</code> families should work through the same generic mechanisms but are untested.
...	Other arguments passed to <code>glm()</code> . Note that <code>weights</code> , <code>subset</code> , and <code>offset</code> don't work reliably here – see Details below.

Details

The returned object has class `c("erglm_model", "glm", "lm")`: it is a `glm` object, with a little extra metadata attached. This means all of the usual `glm/lm` methods work unchanged, without needing an `erglm`-specific equivalent – e.g. `summary()`, `coef()`, `vcov()`, `confint()`, `predict()`, `AIC()`, `BIC()`, `logLik()`, and `anova()` for comparing nested models. See `vignette("methods", package = "erglm")` for worked examples of these. `erglm_predict()` is a separate, `erglm`-specific alternative to `predict()` that returns confidence intervals on the response scale in a tidy data frame; the two are complementary, not competing.

`weights`, `subset`, and `offset` can't currently be passed through `...` to `stats::glm()`: `glm()` resolves these non-standard-evaluation arguments via `match.call()`, which breaks once they've been forwarded through another function's `...` rather than named directly in the call `glm()` itself sees. This reproduces with a trivial wrapper (`function(formula, data, family, ...) glm(formula, data, family, ...)`) and is a limitation of `glm()/lm()`'s NSE, not something specific to `erglm`'s family generalisation – see e.g. the "Note" in `?lm` about wrapping `lm()`. Attempting it currently

fails with a low-level error ("`. . .` used in an incorrect context, no `. . .` to look in"). Two workarounds: fold an offset into the formula itself (e.g. `y ~ x + offset(z)`) rather than passing `offset =`, and pre-filter data yourself rather than passing `subset =`. There's no similar formula-level workaround for weights; call `stats::glm(formula, data, family, weights = ...)` directly instead, then (if you want the `erglm_model` class for consistency, e.g. for `simulate()`'s S3 dispatch) run `class(mod) <- c("erglm_model", class(mod))` on the result – every other `erglm` function works on a plain `glm` object just as well, since none of them require the class specifically.

Value

A `glm` object

Examples

```
mod <- erglm_model(ae1 ~ aucss, erglm_data, family = binomial())
mod

# other glm() families are also supported
mod_pois <- erglm_model(ae_count ~ aucss, erglm_data, family = poisson())
mod_pois
```

erglm_predict

Predictions and confidence intervals for exposure-response models

Description

Computes model-based predictions and confidence intervals on the response scale, returned as a tidy data frame bound to `newdata`.

Usage

```
erglm_predict(object, newdata = NULL, conf_level = 0.95)
```

Arguments

<code>object</code>	An <code>erglm</code> model, as returned by <code>erglm_model()</code>
<code>newdata</code>	Data frame containing cases to be predicted. Defaults to <code>NULL</code> , in which case the data the model was originally fitted to (<code>object\$data</code>) is used.
<code>conf_level</code>	Confidence level for the intervals. Defaults to <code>0.95</code> .

Details

Computes intervals on the link scale and back-transforms with `stats::family(object)$linkinv`, so this works for any `glm()` family, not just binomial/logistic models. See also `erglm_fun()` for generating predictions at arbitrary (possibly counterfactual) parameters or data. `conf_level` must be a single number between 0 and 1 (inclusive); other values error rather than silently producing a reversed or `NaN` interval.

This is a tidy, opinionated alternative to calling base R's `predict()` directly on object – since object is a genuine glm object, `predict()` (and `predict(object, se.fit = TRUE)`, on which this function is based) work unchanged and remain useful for quick point estimates or when a tidy data frame isn't needed. See `vignette("methods", package = "erglm")` for a side-by-side comparison and other inherited glm/lm methods (`summary()`, `vcov()`, `AIC()`, etc.).

Value

A data frame

Examples

```
mod <- erglm_model(ae1 ~ aucss, erglm_data, family = binomial())
prd <- erglm_predict(mod, erglm_data)
head(prd)

mod_gauss <- erglm_model(biomarker_change ~ aucss, erglm_data, family = gaussian())
prd_gauss <- erglm_predict(mod_gauss, erglm_data)
head(prd_gauss)
```

erglm_scm

Stepwise covariate modelling for exposure-response models

Description

Automates the search for which covariates belong in an exposure-response model: `erglm_scm_forward()` greedily adds candidate terms, `erglm_scm_backward()` greedily removes them, and `erglm_scm_history()` retrieves the audit log of every model considered along the way.

Usage

```
erglm_scm_forward(
  mod,
  candidates,
  threshold = 0.01,
  criterion = "p-value",
  test = c("auto", "Chisq", "F"),
  seed = NULL
)

erglm_scm_backward(
  mod,
  candidates,
  threshold = 0.001,
  criterion = "p-value",
  test = c("auto", "Chisq", "F"),
  seed = NULL
)
```

```
)

erglm_scm_history(mod)
```

Arguments

<code>mod</code>	An erglm model object
<code>candidates</code>	Character vector with list of candidate terms
<code>threshold</code>	Threshold to test against. Used only when <code>criterion = "p-value"</code> (the default); ignored otherwise. Defaults to 0.01 for <code>erglm_scm_forward()</code> and 0.001 for <code>erglm_scm_backward()</code> .
<code>criterion</code>	Model selection criterion. One of "p-value" (default), "aic", or "bic".
<code>test</code>	Which significance test to use when comparing nested models. Only used when <code>criterion = "p-value"</code> . "auto" (the default) picks a likelihood-ratio chi-squared test ("Chisq") for families with known dispersion (binomial, poisson) and an F-test ("F") for families with an estimated dispersion parameter (gaussian, gamma, inverse.gaussian, quasi*), matching <code>stats::anova()</code> 's own test argument. Set explicitly to override.
<code>seed</code>	Optional seed controlling the order candidate terms are tested in within a step. Defaults to NULL, in which case one is chosen automatically and used silently – unlike <code>simulate.erglm_model()</code> 's auto-picked seed, it is not reported, since (per Details below) it essentially never changes the result.

Value

For `erglm_scm_forward()` and `erglm_scm_backward()`, the updated erglm model is returned, with the SCM history log updated internally. For `erglm_scm_history()`, a data frame is returned containing the SCM history log

Reproducibility and the seed argument

`seed` exists as a safety measure against two hypothetical sources of run-to-run variation: (a) the order in which candidate terms are tested within a step, and (b) some part of the model-fitting machinery secretly depending on `.Random.seed`. As currently implemented, only (a) is real, and even then its effect is usually invisible. Concretely: each step of `erglm_scm_forward()/erglm_scm_backward()` shuffles the candidate terms (`sample()`) before testing them one at a time, and the shuffled order is the *only* thing `seed` (via a seeded-then-restored RNG block) controls. Term p-values come from `stats::anova()` on models fitted with `stats::glm()`, which is a deterministic algorithm (iteratively reweighted least squares, no random starting values) – so which candidate is *found* to be best does not depend on the seed. The seed can only change which candidate is *selected* in the (rare, essentially measure-zero for continuous predictors) case of an exact tie in p-values within a step, since ties are broken by encounter order (`p_val < lowest_p / p_val > highest_p` are strict inequalities in the internal `.erglm_once_forward()/erglm_once_backward()` helpers). In short: for typical data, `seed` is redundant for reproducibility of the *result* (though it still affects the row order of the intermediate attempts recorded in `erglm_scm_history()`) – it's retained mainly as a guard against future refactors reintroducing genuine seed-sensitivity (e.g. if candidate order were ever used as an early-stopping rule rather than exhaustively tested every step).

Aliased or collinear candidates

If a candidate term is aliased (perfectly collinear) with a term already in the model, `stats::anova()` reports zero additional degrees of freedom and an NA p-value for it. That candidate is skipped for the step (with a warning) rather than being selected or crashing the search – comparisons against NA aren't meaningful, and the candidate can never improve the fit anyway once it's aliased.

Selection criteria

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

- "p-value" (default): Models are compared with the significance test named by `test`. A term is added if its p-value falls below `threshold` (forward) or removed if its p-value exceeds `threshold` (backward). When multiple candidates satisfy the threshold within a step, 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 relative to the current model. 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` and `test` arguments have no effect, and `term_p_value` is left NA in the history for every candidate tested that step (the significance test isn't computed, since it plays no role in selection). The `model_aic` and `model_bic` columns are always recorded regardless of which criterion drove selection, and the history's `criterion` column records which one was used for each forward/backward step.

Candidate validation

`candidates` is validated up front: every element must be parseable as a formula and name exactly one covariate term (e.g. "sex", not "sex + dose" or "not a formula"). This errors immediately, before any model fitting, naming the offending element – rather than surfacing only once the search happens to test that candidate, many steps into what might be a long, expensive search.

Examples

```
mod0 <- erglm_model(ae1 ~ aucss, erglm_data, family = binomial())
mod1 <- erglm_scm_forward(mod0, candidates = c("sex", "dose"))
erglm_scm_history(mod1)

mod2 <- erglm_model(ae1 ~ aucss + sex + dose, erglm_data, family = binomial())
mod3 <- erglm_scm_backward(mod2, candidates = c("sex", "dose"))
erglm_scm_history(mod3)

# AIC-based forward addition/backward elimination instead of p-value
mod4 <- erglm_scm_forward(mod0, candidates = c("sex", "dose"), criterion = "aic")
mod5 <- erglm_scm_backward(mod4, candidates = c("sex", "dose"), criterion = "bic")
erglm_scm_history(mod5)
```

erglm_term

*Add or remove a covariate term from an exposure-response model***Description**

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

Usage

```
erglm_add_term(mod, term, quiet = FALSE)
```

```
erglm_remove_term(mod, term, quiet = FALSE)
```

Arguments

<code>mod</code>	An erglm model object, as returned by erglm_model()
<code>term</code>	A one-sided formula naming the term to add/remove, e.g. <code>~ sex</code>
<code>quiet</code>	If TRUE, suppress the warning issued when the term can't be added/removed (because it's already in the model / isn't in the model, respectively). Defaults to FALSE.

Details

These functions are not typically called directly; they underpin [erglm_scm_forward\(\)](#) and [erglm_scm_backward\(\)](#). Named and shaped to match the companion `emaxnls` package's [emaxnls::emax_add_term\(\)](#)/[emaxnls::emax_remove_term\(\)](#) which serve the same purpose for `emaxnls/emaxlogistic` models – with one structural difference: `emaxnls`'s terms are two-sided formulas naming a structural parameter (e.g. `E0 ~ AGE`), since covariates there attach to a specific Emax parameter, whereas `erglm`'s terms are plain one-sided `glm()` formula terms (e.g. `~ sex`), since `erglm` has no equivalent parameter-level structure to attach covariates to. `term` must be a one-sided formula naming exactly one covariate; passing NULL, a non-formula, a two-sided formula, or a multi-term formula (e.g. `~ weight + age`) errors informatively rather than failing with a low-level error (NULL/non-formula) or being silently misinterpreted (two-sided formulas; multi-term formulas, which used to add/attempt every term at once with no warning).

Value

An erglm model object. If the term can't be added/removed (see `quiet`), the original `mod` is returned unchanged.

Examples

```
mod <- erglm_model(ae1 ~ aucss, erglm_data, family = binomial())
mod2 <- erglm_add_term(mod, ~ sex)
mod3 <- erglm_remove_term(mod2, ~ sex)
```

`simulate.erglm_model` *Simulate responses from an exposure-response model*

Description

Generates simulated response datasets from a fitted `erglm` model, propagating uncertainty in the parameter estimates. Useful for simulation-based confidence bands, predictive checks, or bootstrapping downstream analyses. Implements the standard `stats::simulate()` generic, so it is called as `simulate(object, ...)` rather than through an `erglm`-specific function name.

Usage

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

Arguments

<code>object</code>	An <code>erglm</code> model, as returned by erglm_model()
<code>nsim</code>	Number of replicates. Must be a single positive whole number. Defaults to 1.
<code>seed</code>	Used to set the RNG seed. If <code>NULL</code> , a random seed is chosen and reported.
<code>...</code>	Ignored

Details

Samples new parameter values from the multivariate normal distribution implied by the model's variance-covariance matrix (via `mvtnorm::rmvnorm()`), evaluates the expected response at each sampled parameter vector using [erglm_fun\(\)](#), then draws a simulated response at each prediction using family-appropriate residual noise (the same `.erglm_draw_response()` mechanism used elsewhere in the package: Bernoulli draws for `binomial`, Poisson draws for `poisson`, normal draws for `gaussian`, gamma draws for `Gamma`). The dispersion parameter used for that noise is a single point estimate (`summary(object)$dispersion`), not resampled per replicate. Other `glm()` families are not currently supported and will raise an informative error.

For a VPC-style plot comparing observed and simulated response rates, see the companion `erplots` package's `er_vpc()` mini-grammar – specifically [erplots::er_vpc_add_simulated\(\)](#), which can build its simulated replicates directly from a fitted model (its `model` = argument) via the same `.erglm_draw_response()` noise mechanism this method uses, without needing to call `simulate()` yourself.

Value

A data frame with one row per observation per simulated replicate, containing:

- `dat_id`, `sim_id`: identifiers for the original observation and the simulation replicate
- `mu`: the expected response (response scale) at the sampled parameter vector
- `val`: the simulated response value (`mu` plus family-appropriate noise)

- one `coef_*` column per model coefficient (e.g. `coef_`(Intercept)``, `coef_aucss`), giving the sampled parameter values used for that replicate – prefixed to avoid colliding with predictor columns of the same name
- the model's predictor columns (not including the response)

Examples

```
mod <- erglm_model(ae1 ~ aucss + sex, erglm_data, family = binomial())
sim <- simulate(mod, nsim = 5, seed = 963)
head(sim)
```

Index

*** datasets**

- `erglm_data`, [2](#)
- `emaxnls::emax_add_term()`, [10](#)
- `emaxnls::emax_fun()`, [3](#)
- `emaxnls::emax_remove_term()`, [10](#)
- `erglm_add_term(erglm_term)`, [10](#)
- `erglm_data`, [2](#)
- `erglm_fun`, [3](#)
- `erglm_fun()`, [6](#), [11](#)
- `erglm_invlink(erglm_link)`, [4](#)
- `erglm_link`, [4](#)
- `erglm_model`, [5](#)
- `erglm_model()`, [3](#), [6](#), [10](#), [11](#)
- `erglm_predict`, [6](#)
- `erglm_predict()`, [5](#)
- `erglm_remove_term(erglm_term)`, [10](#)
- `erglm_scm`, [7](#)
- `erglm_scm_backward(erglm_scm)`, [7](#)
- `erglm_scm_backward()`, [10](#)
- `erglm_scm_forward(erglm_scm)`, [7](#)
- `erglm_scm_forward()`, [10](#)
- `erglm_scm_history(erglm_scm)`, [7](#)
- `erglm_scm_history()`, [8](#)
- `erglm_term`, [10](#)
- `erplots::er_vpc_add_simulated()`, [11](#)
- `simulate.erglm_model`, [11](#)
- `simulate.erglm_model()`, [8](#)