Package 'erglm'

August 8, 2026

Title Exposure-Response Tools for GLM-Based Models

Version 0.1.1

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 dplyr, mvtnorm, rlang, stats, tibble, withr

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

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

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

Additional_repositories https://djnavarro.r-universe.dev

Depends R (>= 4.1.0)

LazyData true

Config/testthat/edition 3

Config/Needs/website djnavarro/waeponwifestre, rmarkdown

Config/roxygen2/version 8.0.0

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 2026-08-08 09:00:02 UTC

Contents

`erglm_data` *Sample simulated data for exposure-response models with covariates*

Description

Sample simulated data for exposure-response models with covariates

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 You can find the data generating code in the package source code

Examples

```
erglm_data
```

`erglm_fun` *Prediction function for an exposure-response model*

Description

Prediction function for an exposure-response model

Usage

```
erglm_fun(object)
```

Arguments

`object`	An erglm model, as returned by `erglm_model()`

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"`)

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

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 *Fit an exposure-response model based on* glm()

Description

Fit an exposure-response model based on glm()

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 stats::glm(). Defaults to stats::gaussian(), matching stats::glm()'s own default. Tested and officially supported for binomial(), poisson(), gaussian(), and Gamma(); other glm() families should work through the same generic mechanisms but are untested.
...	Other arguments passed to glm(). Note that weights, subset, and offset 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 ("..1 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

Predictions and confidence intervals for exposure-response models

Usage

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

Arguments

object	An erglm model, as returned by erglm_model()
newdata	Data frame containing cases to be predicted
conf_level	Confidence level for the intervals

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 tibble

Examples

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

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

`erglm_scm` *Stepwise covariate modelling for exposure-response models*

Description

Stepwise covariate modelling for exposure-response models

Usage

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

erglm_scm_backward(
  mod,
  candidates,
  threshold = 0.001,
  test = c("auto", "Chisq", "F"),
  seed = NULL
)

erglm_scm_history(mod)
```

Arguments

`mod`	An erglm model object
`candidates`	Character vector with list of candidate terms
`threshold`	Threshold to test against
`test`	Which significance test to use when comparing nested models. `"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 `stats::anova()`'s own `test` argument. Set explicitly to override.
`seed`	Optional seed to control order of term tests

Details

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 withr::with_seed()) 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).

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.

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.

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

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)
```

---

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

mod	An erglm model object, as returned by erglm_model()
term	A one-sided formula naming the term to add/remove, e.g. ~ sex
quiet	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)

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 emax_add_term()/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

`object`	An erglm model, as returned by `erglm_model()`
`nsim`	Number of replicates. Must be a single positive whole number.
`seed`	Used to set the RNG seed. If `NULL`, a random seed is chosen and reported.
`...`	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_plot()`, which can build its simulated replicates directly from a fitted model (`model =` argument) via the same `.erglm_draw_response()` noise mechanism this method uses, without needing to call `simulate()` yourself.

Value

A tibble 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())
simulate(mod, nsim = 5, seed = 963)
```

Index