Package ‘BayesERtools’

February 12, 2025

Type Package

Title Bayesian Exposure-Response Analysis Tools

Version 0.2.1

Maintainer Kenta Yoshida <yoshida.kenta.6@gmail.com>

Description Suite of tools that facilitate
exposure-response analysis using Bayesian methods. The package
provides a streamlined workflow for fitting types of models that are
commonly used in exposure-response analysis - linear and Emax for continuous
endpoints, logistic linear and logistic Emax for binary endpoints, as well
as performing simulation and visualization. Learn more about the workflow
at <https://genentech.github.io/BayesERbook/>.

License Apache License 2.0

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

URL https://genentech.github.io/BayesERtools/,
https://genentech.github.io/BayesERbook/

Config/testthat/edition 3

RoxygenNote 7.3.2

Imports dplyr, tidyr, purrr, ggplot2, gt, cli, rlang, rstanarm,
rstanemax (>= 0.1.8), loo, tidybayes, bayestestR, posterior

Suggests testthat (>= 3.0.0), covr, knitr, rmarkdown, rstan,
htmltools, digest, ggforce, xgxr, scales, readr, patchwork,
projpred, rsample, yardstick

VignetteBuilder knitr

NeedsCompilation no

Author Kenta Yoshida [aut, cre] (<https://orcid.org/0000-0003-4967-3831>),
François Mercier [aut] (<https://orcid.org/0000-0002-5685-1408>),
Genentech, Inc. [cph]

Repository CRAN

Date/Publication 2025-02-12 11:40:22 UTC

Contents

as_draws 2
build_spec_coveff 3
calc_ersim_med_qi 5
dev_ermod_bin 6
dev_ermod_bin_cov_sel 7
dev_ermod_bin_exp_sel 9
dev_ermod_emax 11
dev_ermod_emax_exp_sel 13
d_sim_binom_cov 15
d_sim_lin 16
edit_spec_coveff 17
ermod_cov_sel_method 19
ermod_exp_sel_method 19
ermod_method 20
ersim_method 21
eval_ermod 21
extract_coef_exp_ci 23
extract_method 23
loo 24
plot_coveff 25
plot_cov_sel 26
plot_er 27
plot_er_exp_sel 30
plot_er_gof 31
print_coveff 33
p_direction 34
run_kfold_cv 36
sim_coveff 37
sim_er 38
sim_er_new_exp 39
sim_er_new_exp_marg 42

Index **45**

`as_draws` *Transform to* `draws` *objects*

Description

See `posterior::as_draws()` for details.

Usage

```
as_draws(x, ...)

as_draws_list(x, ...)

as_draws_array(x, ...)

as_draws_df(x, ...)

as_draws_matrix(x, ...)

as_draws_rvars(x, ...)

## S3 method for class 'ermod'
as_draws(x, ...)

## S3 method for class 'ermod'
as_draws_list(x, ...)

## S3 method for class 'ermod'
as_draws_array(x, ...)

## S3 method for class 'ermod'
as_draws_df(x, ...)

## S3 method for class 'ermod'
as_draws_matrix(x, ...)

## S3 method for class 'ermod'
as_draws_rvars(x, ...)
```

Arguments

`x`	An object of class `ermod`
`...`	Arguments passed to individual methods (if applicable).

Value

A draws object from the `posterior` package.

`build_spec_coveff` *Build specifications for covariate effect simulation/visualization*

Description

Build specifications for covariate effect simulation/visualization

Usage

```
build_spec_coveff(
  ermod,
  data = NULL,
  qi_width_cov = 0.9,
  n_sigfig = 3,
  use_seps = TRUE,
  drop_trailing_dec_mark = TRUE
)
```

Arguments

`ermod` an object of class `ermod`

`data` an optional data frame to derive the covariate values for forest plots. If NULL (default), the data used to fit the model is used.

`qi_width_cov` the width of the quantile interval for continuous covariates in the forest plot. Default is 0.9 (i.e. visualize effect of covariate effect at their 5th and 95th percentile values).

`n_sigfig` Number of significant figures to form value_label of continuous variables. See `gt::vec_fmt_number()` for details.

`use_seps` Whether to use separators for thousands in printing numbers. See `gt::vec_fmt_number()` for details.

`drop_trailing_dec_mark`
Whether to drop the trailing decimal mark (".") in value_label of continuous variables. See `gt::vec_fmt_number()` for details.

Value

`spec_coveff` (return object) is a data frame for the specification of the covariate effects to be visualized. This is internally generated by `build_spec_coveff()` if you run `sim_coveff()` or `plot_coveff()` directly. Alternatively, you can develop your own or modify the one generated by `build_spec_coveff()` and supply it to `sim_coveff()` or `plot_coveff()`. The data frame should have the following columns (but it's probably easier to try `build_spec_coveff()` and see the structure):

- `var_order`: The order of the covariate in the forest plot. The exposure variable is always the first one and the covariates are ordered by the order they are supplied in the `var_cov` argument of the `dev_ermod_*` function. If you used a model from `dev_ermod_bin_cov_sel()`, then the order is determined by the variable selection process.
- `var_name`: The name of the variable.
- `var_label`: The label of the variable to be used for plot. This is the same as `var_name` by default.
- `value_order`: The order of the value of the variable to be evaluated.
- `value_annot`: The annotation of the value of the variable to be evaluated. This appears on the right hand side of the forest plot.
- `value_label`: The label of the value of the variable to be evaluated.

- `value_cont`: The value for continuous variables.
- `value_cat`: The value for categorical variables.
- `is_ref_value`: Whether the value is the reference value.
- `show_ref_value`: Whether to show the reference value in the plot and table. This is TRUE by default for is_ref_value == TRUE, otherwise NA (and ignored).
- `is_covariate`: Whether the variable is a covariate (TRUE) or exposure variable (FALSE).

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = c("BHBA1C_5", "RACE"),
)

spec_coveff <- build_spec_coveff(ermod_bin)
plot_coveff(ermod_bin, spec_coveff = spec_coveff)
```

`calc_ersim_med_qi` *Calculate median and quantile intervals from ersim object*

Description

This is useful when you performed simulation with output_type = "draws" and want to calculate median and quantile intervals without re-simulating.

Usage

```
calc_ersim_med_qi(x, qi_width = 0.95)
```

Arguments

`x`	An object of class `ersim` or `ersim_marg`
`qi_width`	Width of the quantile interval

Value

An object of class `ersim_med_qi` or `ersim_marg_med_qi`

`dev_ermod_bin` *Develop linear ER model for binary or continuous endpoint*

Description

These functions are used to develop an linear ER model with binary (`dev_ermod_bin()`) or continuous (`dev_ermod_lin()`) endpoint. You can also specify covariates to be included in the model.

Usage

```
dev_ermod_bin(
  data,
  var_resp,
  var_exposure,
  var_cov = NULL,
  verbosity_level = 1,
  chains = 4,
  iter = 2000
)

dev_ermod_lin(
  data,
  var_resp,
  var_exposure,
  var_cov = NULL,
  verbosity_level = 1,
  chains = 4,
  iter = 2000
)
```

Arguments

`data`	Input data for E-R analysis
`var_resp`	Response variable name in character
`var_exposure`	Exposure variable names in character
`var_cov`	Covariate variable names in character vector
`verbosity_level`	Verbosity level. 0: No output, 1: Display steps, 2: Display progress in each step, 3: Display MCMC sampling.
`chains`	Number of chains for Stan.
`iter`	Number of iterations for Stan.

Value

An object of class `ermod_bin` or `ermod_lin`.

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

ermod_bin



data(d_sim_lin)

ermod_lin <- dev_ermod_lin(
  data = d_sim_lin,
  var_resp = "response",
  var_exposure = "AUCss",
  var_cov = c("SEX", "BAGE")
)

ermod_lin
```

`dev_ermod_bin_cov_sel` *Perform covariate selection for linear ER model*

Description

This functions is used to develop an ER model with covariates for binary and continuous endpoints. `projpred` package is used for variable selection.

Usage

```
dev_ermod_bin_cov_sel(
  data,
  var_resp,
  var_exposure,
  var_cov_candidates,
  cv_method = c("LOO", "kfold"),
  k = 5,
  validate_search = FALSE,
  nterms_max = NULL,
  .reduce_obj_size = TRUE,
  verbosity_level = 1,
  chains = 4,
  iter = 2000
)

dev_ermod_lin_cov_sel(
  data,
  var_resp,
  var_exposure,
  var_cov_candidates,
  cv_method = c(”LOO”, ”kfold”),
  k = 5,
  validate_search = FALSE,
  nterms_max = NULL,
  .reduce_obj_size = TRUE,
  verbosity_level = 1,
  chains = 4,
  iter = 2000
)
```

Arguments

`data` Input data for E-R analysis

`var_resp` Response variable name in character

`var_exposure` Exposure variable names in character

`var_cov_candidates` Candidate covariate names in character vector

`cv_method` Cross-validation method. Default is "LOO" (recommended). Use "kfold" if you see warnings on Pareto k estimates.

`k` Number of folds for kfold CV. Only used if cv_method is "kfold".

`validate_search` Whether to validate the search. Default is FALSE. Recommend to set to TRUE for kfold CV. Do not use for LOO (run time would become too long).

`nterms_max` Maximum number of terms to consider in the model. Default is NULL (all terms are considered).

`.reduce_obj_size` Whether to reduce object size by removing some elements from projpred outputs that are not necessary for the functionality of this package.

`verbosity_level` Verbosity level. 0: No output, 1: Display steps, 2: Display progress in each step, 3: Display MCMC sampling.

`chains` Number of chains for Stan.

`iter` Number of iterations for Stan.

Value

An object of class `ermod_bin_cov_sel` or `ermod_lin_cov_sel`.

Examples

```
data(d_sim_binom_cov_hgly2)

er_binary_cov_model <- dev_ermod_bin_cov_sel(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov_candidates = c(
    "BAGE_10", "BWT_10", "BGLUC",
    "BHBA1C_5", "RACE", "VISC"
  )
)

er_binary_cov_model




data(d_sim_lin)

ermod_lin_cov_sel <- dev_ermod_lin_cov_sel(
  data = d_sim_lin,
  var_resp = "response",
  var_exposure = "AUCss",
  var_cov_candidates = c("BAGE", "SEX")
)

ermod_lin_cov_sel
```

dev_ermod_bin_exp_sel *Exposure metrics selection for linear ER models*

Description

This functions is used to develop an linear ER model with binary and continuous endpoint, using various exposure metrics and selecting the best one.

Usage

```
dev_ermod_bin_exp_sel(
  data,
  var_resp,
  var_exp_candidates,
  verbosity_level = 1,
  chains = 4,
  iter = 2000
)

dev_ermod_lin_exp_sel(
  data,
  var_resp,
  var_exp_candidates,
  verbosity_level = 1,
  chains = 4,
  iter = 2000
)
```

Arguments

`data` Input data for E-R analysis

`var_resp` Response variable name in character

`var_exp_candidates`
Candidate exposure variable names in character vector

`verbosity_level`
Verbosity level. 0: No output, 1: Display steps, 2: Display progress in each step, 3: Display MCMC sampling.

`chains` Number of chains for Stan.

`iter` Number of iterations for Stan.

Value

An object of class `ermod_bin_exp_sel`.or `ermod_lin_exp_sel`

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin_exp_sel <-
  dev_ermod_bin_exp_sel(
    data = d_sim_binom_cov_hgly2,
    var_resp = "AEFLAG",
    var_exp_candidates = c("AUCss_1000", "Cmaxss", "Cminss")
  )

ermod_bin_exp_sel



data(d_sim_lin)

ermod_lin_exp_sel <- dev_ermod_lin_exp_sel(
  data = d_sim_lin,
  var_resp = "response",
  var_exp_candidates = c("AUCss", "Cmaxss")
)
```

ermod_lin_exp_sel

dev_ermod_emax *Develop Emax model for continuous and binary endpoint*

Description

These functions are used to develop an Emax model with continuous or binary endpoint. You can also specify covariates to be included in the model; note that only categorical covariates are allowed.

Usage

```
dev_ermod_emax(
  data,
  var_resp,
  var_exposure,
  l_var_cov = NULL,
  gamma_fix = 1,
  e0_fix = NULL,
  emax_fix = NULL,
  priors = NULL,
  verbosity_level = 1,
  chains = 4,
  iter = 2000,
  seed = sample.int(.Machine$integer.max, 1)
)

dev_ermod_bin_emax(
  data,
  var_resp,
  var_exposure,
  l_var_cov = NULL,
  gamma_fix = 1,
  e0_fix = NULL,
  emax_fix = NULL,
  priors = NULL,
  verbosity_level = 1,
  chains = 4,
  iter = 2000,
  seed = sample.int(.Machine$integer.max, 1)
)
```

Arguments

data	Input data for E-R analysis

`var_resp` Response variable name in character

`var_exposure` Exposure variable names in character

`l_var_cov` a names list of categorical covariate variables in character vector. See details in the `param.cov` argument of `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

`gamma_fix` Hill coefficient, default fixed to 1. See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

`e0_fix` See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

`emax_fix` See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

`priors` See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

`verbosity_level`
Verbosity level. 0: No output, 1: Display steps, 2: Display progress in each step, 3: Display MCMC sampling.

`chains` Number of chains for Stan.

`iter` Number of iterations for Stan.

`seed` Random seed for Stan model execution, see details in `rstan::sampling()` which is used in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

Value

An object of class `ermod_emax`.or `ermod_bin_emax`.

Examples

```
data_er_cont <- rstanemax::exposure.response.sample

ermod_emax <-
  dev_ermod_emax(
    data = data_er_cont,
    var_exposure = "exposure",
    var_resp = "response"
  )

plot_er(ermod_emax, show_orig_data = TRUE)

data_er_cont_cov <- rstanemax::exposure.response.sample.with.cov

ermod_emax_w_cov <-
  dev_ermod_emax(
    data = data_er_cont_cov,
    var_exposure = "conc",
    var_resp = "resp",
    l_var_cov = list(emax = "cov2", ec50 = "cov3", e0 = "cov1")
  )
```

```
data_er_bin <- rstanemax::exposure.response.sample.binary

ermod_bin_emax <-
  dev_ermod_bin_emax(
    data = data_er_bin,
    var_exposure = "conc",
    var_resp = "y"
  )

plot_er(ermod_bin_emax, show_orig_data = TRUE)

ermod_bin_emax_w_cov <-
  dev_ermod_bin_emax(
    data = data_er_bin,
    var_exposure = "conc",
    var_resp = "y_cov",
    l_var_cov = list(emax = "sex")
  )
```

`dev_ermod_emax_exp_sel` *Exposure metrics selection for Emax models*

Description

This functions is used to develop an Emax model with binary and continuous endpoint, using various exposure metrics and selecting the best one.

Usage

```
dev_ermod_emax_exp_sel(
  data,
  var_resp,
  var_exp_candidates,
  verbosity_level = 1,
  chains = 4,
  iter = 2000,
  gamma_fix = 1,
  e0_fix = NULL,
  emax_fix = NULL,
  priors = NULL,
  seed = sample.int(.Machine$integer.max, 1)
)

dev_ermod_bin_emax_exp_sel(
  data,
  var_resp,
  var_exp_candidates,
  verbosity_level = 1,
  chains = 4,
  iter = 2000,
  gamma_fix = 1,
  e0_fix = NULL,
  emax_fix = NULL,
  priors = NULL,
  seed = sample.int(.Machine$integer.max, 1)
)
```

Arguments

`data`	Input data for E-R analysis
`var_resp`	Response variable name in character
`var_exp_candidates`	Candidate exposure variable names in character vector
`verbosity_level`	Verbosity level. 0: No output, 1: Display steps, 2: Display progress in each step, 3: Display MCMC sampling.
`chains`	Number of chains for Stan.
`iter`	Number of iterations for Stan.
`gamma_fix`	Hill coefficient, default fixed to 1. See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`
`e0_fix`	See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`
`emax_fix`	See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`
`priors`	See details in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`
`seed`	Random seed for Stan model execution, see details in `rstan::sampling()` which is used in `rstanemax::stan_emax()` or `rstanemax::stan_emax_binary()`

Value

An object of class `ermod_emax_exp_sel` or `ermod_bin_emax_exp_sel`.

Examples

```
data_er_cont <- rstanemax::exposure.response.sample
noise <- 1 + 0.5 * stats::rnorm(length(data_er_cont$exposure))
data_er_cont$exposure2 <- data_er_cont$exposure * noise
# Replace exposure < 0 with 0
data_er_cont$exposure2[data_er_cont$exposure2 < 0] <- 0

ermod_emax_exp_sel <-
  dev_ermod_emax_exp_sel(
    data = data_er_cont,
    var_resp = "response",
    var_exp_candidates = c("exposure", "exposure2")
  )

ermod_emax_exp_sel



data_er_bin <- rstanemax::exposure.response.sample.binary

noise <- 1 + 0.5 * stats::rnorm(length(data_er_bin$conc))
data_er_bin$conc2 <- data_er_bin$conc * noise
data_er_bin$conc2[data_er_bin$conc2 < 0] <- 0

ermod_bin_emax_exp_sel <-
  dev_ermod_bin_emax_exp_sel(
    data = data_er_bin,
    var_resp = "y",
    var_exp_candidates = c("conc", "conc2")
  )
```

d_sim_binom_cov *Sample simulated data for exposure-response with binary endpoint.*

Description

Sample simulated data for exposure-response with binary endpoint.

Usage

```
d_sim_binom_cov

d_sim_binom_cov_hgly2
```

Format

A data frame with columns:

ID Subject ID

AETYPE Adverse event type: hgly2 (Gr2+ hyperglycemia), dr2 (Gr2+ Diarrhea), ae_covsel_test (hypothetical AE for covariate selection function test)

AEFLAG Adverse event flag: 0 - no event, 1 - event

Dose_mg Dose in mg: 200, 400

AUCss Steady-state area under the curve

Cmaxss Steady-state maximum (peak) concentration

Cminss Steady-state minimum (trough) concentration

BAGE Baseline age in years

BWT Baseline weight in kg

BGLUC Baseline glucose in mmol/L

BHBA1C Baseline HbA1c in percentage

RACE Race: White, Black, Asian

VISC Visceral disease: No, Yes

AUCss_1000 AUCss/1000

BAGE_10 BAGE/10

BWT_10 BWT/10

BHBA1C_5 BHBA1C/5

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 500 rows and 17 columns.

Details

This simulated dataset is very loosely inspired by ER analysis of ipatasertib by Kotani (2022) at:

https://doi.org/10.1007/s00280-022-04488-2

You can find the data generating code in the package source code, under `data-raw/d_sim_binom_cov.R`.

d_sim_binom_cov_hgly2 is a subset of this dataset with only hgly2 AE type and some columns added for testing.

Examples

```
d_sim_binom_cov
d_sim_binom_cov_hgly2
```

`d_sim_lin` *Sample simulated data for exposure-response with continuous endpoint using linear model.*

Description

Sample simulated data for exposure-response with continuous endpoint using linear model.

Usage

```
d_sim_lin
```

Format

A data frame with columns:

ID Subject ID

AUCss Steady-state area under the curve

Cmaxss Steady-state maximum (peak) concentration

BAGE Baseline age in years

SEX M or F

response Response

Details

True model is defined as `0.5 * AUCss + 0.5 * BAGE + 5 * SEX`, with variability added with standard deviation of 10. You can find the data generating code in the package source code, under `data-raw/d_sim_lin.R`.

Examples

```
d_sim_lin
```

`edit_spec_coveff` *Customize specifications for covariate effect simulations/visualizations*

Description

- `build_spec_coveff_one_variable()` is a helper function to create a new specification for a single variable. This is useful when you want to customize the specification for a single variable.
- `replace_spec_coveff()` is used to replace the specification for some (or all) variables in the original specification data frame. If you want to replace multiple variables, you can just stack the specifications together.

Usage

```
build_spec_coveff_one_variable(
  var_name,
  values_vec,
  qi_width_cov = 0.9,
  n_sigfig = 3,
  use_seps = TRUE,
  drop_trailing_dec_mark = TRUE,
  show_ref_value = TRUE
)

replace_spec_coveff(spec_orig, spec_new, replace_ref_value = FALSE)
```

Arguments

`var_name` The name of the variable for which a new spec is to be created.

`values_vec` The vector of the values for creating a new spec.

`qi_width_cov` the width of the quantile interval for continuous covariates in the forest plot. Default is 0.9 (i.e. visualize effect of covariate effect at their 5th and 95th percentile values).

`n_sigfig` Number of significant figures to form value_label of continuous variables. See `gt::vec_fmt_number()` for details.

`use_seps` Whether to use separators for thousands in printing numbers. See `gt::vec_fmt_number()` for details.

`drop_trailing_dec_mark`
Whether to drop the trailing decimal mark (".") in value_label of continuous variables. See `gt::vec_fmt_number()` for details.

`show_ref_value` Whether to show the reference value in the plot and table. Setting this results in the `show_ref_value` column in the specification data frame.

`spec_orig` Original specification data frame.

`spec_new` New specification data frame. It can be generated by `build_spec_coveff_one_variable()` or manually crafting with the following variables: `var_name`, `var_label`, `value_order`, `value_annot`, `value_label`, `value_cont` or `value_cat`, `is_ref_value`, `show_ref_value`. You can have multiple variables stacked together.

`replace_ref_value`
Whether to replace the reference values from the original specification data frame. Default is FALSE; in this case, show_ref_value is set to FALSE as it can be confusing. If you set replace_ref_value to TRUE, the reference calculation for the forest plot is also done with the one in spec_new.

Value

See `build_spec_coveff()` for the structure of the return object. `build_spec_coveff_one_variable()` returns a data frame corresponding to the specification for a single variable, which can be used as an input to `replace_spec_coveff()`.

Examples

```
set.seed(1234)
data(d_sim_binom_cov_hgly2)

ermod_bin <- suppressWarnings(dev_ermod_bin(
  data = d_sim_binom_cov_hgly2, var_resp = "AEFLAG",
  var_exposure = "AUCss_1000", var_cov = c("BGLUC", "RACE"),
  verbosity_level = 0,
  # Below option to make the example run fast
  chains = 2, iter = 1000
))

spec_coveff <- build_spec_coveff(ermod_bin)
```

```
spec_new_bgluc <- build_spec_coveff_one_variable(
  "BGLUC", seq(4, 8, by = 0.1),
  qi_width_cov = 0.8, show_ref_value = FALSE
)
spec_coveff_new <- replace_spec_coveff(spec_coveff, spec_new_bgluc)
plot_coveff(ermod_bin, spec_coveff = spec_coveff_new)
```

`ermod_cov_sel_method` *S3 methods for the classes* `ermod_bin_cov_sel`

Description

S3 methods for the classes `ermod_bin_cov_sel`

Usage

```
## S3 method for class 'ermod_cov_sel'
print(x, digits = 2, ...)

## S3 method for class 'ermod_cov_sel'
plot(x, ...)
```

Arguments

`x`	An object of class `ermod_bin_cov_sel`
`digits`	Number of digits to print
`...`	Additional arguments passed to functions

Value

No return value, called for print or plot side effects

`ermod_exp_sel_method` *S3 methods for the classes* `ermod_exp_sel`

Description

S3 methods for the classes `ermod_exp_sel`

Usage

```
## S3 method for class 'ermod_exp_sel'
print(x, digits = 2, ...)

## S3 method for class 'ermod_exp_sel'
plot(x, ...)
```

Arguments

`x`	An object of class `ermod_bin_exp_sel`
`digits`	Number of digits to print
`...`	Additional arguments passed to functions

Value

No return value, called for print or plot side effects

`ermod_method` *S3 methods for the classes* `ermod_*`

Description

S3 methods for the classes `ermod_*`

Usage

```
## S3 method for class 'ermod'
print(x, digits = 2, ...)

## S3 method for class 'ermod_bin'
plot(x, show_orig_data = FALSE, ...)

## S3 method for class 'ermod'
coef(object, ...)

## S3 method for class 'ermod'
summary(object, ...)
```

Arguments

`x`	An object of class `ermod_*`
`digits`	Number of digits to print
`...`	Additional arguments passed to functions
`show_orig_data`	logical, whether to show the data points in the model development dataset. Default is `FALSE`. Only support plotting with data that was used in the model development. If you want to use other data, consider adding geom_point() to the plot manually.
`object`	An object of class `ermod_*`

Value

- `print()` and `plot()`: No return value, called for side effects
- `coef()`: Coefficients of the model
- `summary()`: Summary of the model

ersim_method *S3 methods for the classes* ersim_* *and* ersim_med_qi_*

Description

S3 methods for the classes ersim_* and ersim_med_qi_*

Usage

```
## S3 method for class 'ersim'
plot(x, show_orig_data = FALSE, ...)

## S3 method for class 'ersim_med_qi'
plot(x, show_orig_data = FALSE, ...)
```

Arguments

x	An object of the classes ersim_* or ersim_med_qi_*
show_orig_data	logical, whether to show the data points in the model development dataset. Default is FALSE. Only support plotting with data that was used in the model development. If you want to use other data, consider adding geom_point() to the plot manually.
...	Additional arguments passed to functions

Value

No return value, called for print or plot side effects

eval_ermod *Evaluate exposure-response model prediction performance*

Description

This function evaluates the performance of an exposure-response model using various metrics.

Usage

```
eval_ermod(
  ermod,
  eval_type = c("training", "kfold", "test"),
  newdata = NULL,
  summary_method = c("median", "mean"),
  k = 5,
  seed_kfold = NULL
)
```

Arguments

`ermod` An object of class `ermod`.

`eval_type` A character string specifying the evaluation dataset. Options are:

- `training`: Use the training dataset.
- `test`: Use a new dataset for evaluation.
- `kfold`: Perform k-fold cross-validation (uses `newdata` if provided, otherwise uses the training dataset).

`newdata` A data frame containing new data for evaluation when `eval_type` is set to `test` or `kfold`.

`summary_method` A character string specifying how to summarize the simulation draws. Default is `median`.

`k` The number of folds for cross-validation. Default is 5.

`seed_kfold` Random seed for k-fold cross-validation.

Value

A tibble with calculated performance metrics, such as AUROC or RMSE, depending on the model type.

Examples

```
data(d_sim_binom_cov_hgly2)
d_split <- rsample::initial_split(d_sim_binom_cov_hgly2)
d_train <- rsample::training(d_split)
d_test <- rsample::testing(d_split)

ermod_bin <- dev_ermod_bin(
  data = d_train,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
  # Settings to make the example run faster
  chains = 2,
  iter = 1000
)

metrics_training <- eval_ermod(ermod_bin, eval_type = "training")
metrics_test <- eval_ermod(ermod_bin, eval_type = "test", newdata = d_test)
metrics_kfold <- eval_ermod(ermod_bin, eval_type = "kfold", k = 3)

print(metrics_training)
print(metrics_test)
print(metrics_kfold)
```

extract_coef_exp_ci *Extract credible interval of the exposure coefficient*

Description

Extract credible interval of the exposure coefficient

Usage

```
extract_coef_exp_ci(x, ci_width = 0.95)
```

Arguments

`x`	An object of class `ermod_bin` or `ermod_lin`
`ci_width`	Width of the credible interval

Value

A named vector of length 2 with the lower and upper bounds of the credible interval (.lower, .upper)

extract_method *Extract elements from S3 objects*

Description

S3 methods are defined for `ermod_*` (see extract_ermod) and `ersim_*` (see extract_ersim) classes.

Usage

```
extract_data(x)

extract_mod(x)

extract_var_resp(x)

extract_var_exposure(x)

extract_var_cov(x)

extract_exp_sel_list_model(x)

extract_exp_sel_comp(x)

extract_var_selected(x)
```

Arguments

x	An object to extract elements from

Value

- `extract_data()` extracts data used for the model fit.
- `extract_mod()` extracts the model fit object.
- `extract_var_resp()` extracts the response variable name
- `extract_var_exposure()` extracts the exposure metric name
- `extract_var_cov()` extracts the covariates name
- `extract_exp_sel_list_model()` extracts the list of fitted models for each exposure metrics.
- `extract_exp_sel_comp()` extracts the comparison results of the exposure metrics.
- `extract_var_selected()` extracts the selected variables (both exposure and covariates)in the final model after covariate selection.

loo *Efficient approximate leave-one-out cross-validation (LOO)*

Description

See `loo::loo()` for details.

Usage

```
loo(x, ...)

## S3 method for class 'ermod'
loo(x, ...)

## S3 method for class 'ermod_emax'
loo(x, ...)

## S3 method for class 'ermod_bin_emax'
loo(x, ...)
```

Arguments

x	An object of class `ermod`
...	Additional arguments passed to `loo::loo()`

Value

An object of class `loo`

`plot_coveff` *Visualize the covariate effects for ER model*

Description

Visualize the covariate effects for ER model

Usage

```
plot_coveff(x, ...)

## S3 method for class 'ermod'
plot_coveff(
  x,
  data = NULL,
  spec_coveff = NULL,
  qi_width = 0.9,
  qi_width_cov = 0.9,
  ...
)

## S3 method for class 'coveffsim'
plot_coveff(x, ...)
```

Arguments

`x`	an object of class `ermod`, `coveffsim`, or their subclasses
`...`	currently not used
`data`	an optional data frame to derive the covariate values for forest plots. If NULL (default), the data used to fit the model is used.
`spec_coveff`	you can supply spec_coveff to `sim_coveff()` or `plot_coveff()`, if you have already built it manually or with `build_spec_coveff()`. See `build_spec_coveff()` for detail.
`qi_width`	the width of the credible interval on the covariate effect. This translate to the width of the error bars in the forest plot.
`qi_width_cov`	the width of the quantile interval for continuous covariates in the forest plot. Default is 0.9 (i.e. visualize effect of covariate effect at their 5th and 95th percentile values).

Value

A ggplot object

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

plot_coveff(ermod_bin)
```

---

plot_cov_sel *Plot variable selection performance*

Description

Plot variable selection performance

Usage

```
plot_submod_performance(x)

plot_var_ranking(x)
```

Arguments

x	An object of class `ermod_bin_cov_sel`

Details

`plot_submod_performance()` plots the performance of submodels evaluated during variable selection.

`plot_var_ranking()` plots the variable ranking evaluated during variable selection.

Value

No return value, called for plotting side effect.

Examples

```
data(d_sim_binom_cov_hgly2)

er_binary_cov_model_kfold <- dev_ermod_bin_cov_sel(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov_candidate = c(
    "BAGE_10", "BWT_10", "BGLUC",
    "BHBA1C_5", "RACE", "VISC"
  ),
  cv_method = "kfold",
  k = 3, # Choose 3 to make the example go fast
  validate_search = TRUE,
)

plot_submod_performance(er_binary_cov_model_kfold)
plot_var_ranking(er_binary_cov_model_kfold)
```

---

plot_er *Plot ER model simulations*

Description

Plot ER model simulations

Usage

```
plot_er(x, ...)

## S3 method for class 'ersim_med_qi'
plot_er(
  x,
  show_orig_data = FALSE,
  show_coef_exp = FALSE,
  show_caption = FALSE,
  options_orig_data = list(),
  options_coef_exp = list(),
  options_caption = list(),
  ...
)

## S3 method for class 'ersim'
plot_er(
  x,
  show_orig_data = FALSE,
  show_coef_exp = FALSE,
  show_caption = FALSE,
  options_orig_data = list(),
  options_coef_exp = list(),
  options_caption = list(),
  qi_width_sim = 0.95,
  ...
)

## S3 method for class 'ermod'
plot_er(
  x,
  show_orig_data = FALSE,
  show_coef_exp = FALSE,
  show_caption = FALSE,
  options_orig_data = list(),
  options_coef_exp = list(),
  options_caption = list(),
  n_draws_sim = if (marginal) 200 else NULL,
  seed_sample_draws = NULL,
  marginal = FALSE,
  exposure_range = NULL,
  num_exposures = 51,
  qi_width_sim = 0.95,
  ...
)
```

Arguments

`x` an object of class `ermod`, `ersim`,`ersim_med_qi`, or their subclasses

`...` currently not used

`show_orig_data` logical, whether to show the data points in the model development dataset. Default is `FALSE`. Only support plotting with data that was used in the model development. If you want to use other data, consider adding geom_point() to the plot manually.

`show_coef_exp` logical, whether to show the credible interval of the exposure coefficient. Default is `FALSE`. This is only available for linear and linear logistic regression models.

`show_caption` logical, whether to show the caption note for the plot. Default is `FALSE`.

`options_orig_data`

List of options for configuring how original data is displayed. Possible options include:

- `add_boxplot`: Logical, whether to add a boxplot of exposure values. Default is `FALSE`.
- `boxplot_height`: Height of the boxplot relative to the main plot. Default is `0.15`.

- show_boxplot_y_title: Logical, whether to show the y-axis title for the boxplot. Default is TRUE.
- var_group: The column to use for grouping data for plotting. If specified, observed data points and boxplot will be grouped and colored by this column. Default is NULL.
- n_bins: Number of bins to use for observed probability summary. Only relevant for binary models. Default is 4.
- qi_width: Width of the quantile interval (confidence interval) for the observed probability summary. Only relevant for binary models. Default is 0.95.

options_coef_exp

List of options for configuring how the exposure coefficient credible interval is displayed. Possible options include:

- qi_width: Width of the quantile interval (credible interval) for the exposure coefficient. Default is 0.95.
- n_sigfig: Number of significant figures to display. Default is 3.
- pos_x: x-coordinate of the text label. If NULL (default), it is set to the minimum value for the exposure variable.
- pos_y: y-coordinate of the text label. If NULL (default), it is set to 0.9 for logistic regression models and the maximum value of the response variable in the original data for linear regression models.
- size: Size of the text label. Default is 4.

options_caption

List of options for configuring the caption note. Possible options include:

- orig_data: Logical, whether to show the caption note for the observed data. Default is FALSE.
- orig_data_summary: Logical, whether to show the caption note for the observed data summary. Default is FALSE. Only relevant for binary models.
- coef_exp: Logical, whether to show the caption note for the exposure coefficient credible interval. Default is FALSE.

qi_width_sim Width of the quantile interval to summarize simulated draws.

n_draws_sim Number of draws to simulate response for each exposure value. Set to NULL to use all draws in the model object. Default is NULL unless marginal is set to TRUE (in that case 200 by default to reduce computation time).

seed_sample_draws

Seed for sampling draws. Default is NULL.

marginal logical, whether to use marginal ER simulation. Default to FALSE. Need to set to TRUE if the model has covariates for the plot to work.

exposure_range Only relevant when the input x is an ermod object. Range of exposure values to simulate. If NULL (default), it is set to the range of the exposure variable in the original data for model development.

num_exposures Only relevant as with exposure_range. Number of exposure values to simulate.

Details

Plotting with ermod is done with some default values. If they are not suitable, you can always perform the simulation manually and use plot_er() on the simulated data.

Value

A ggplot object

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000"
)

ersim_med_qi <- sim_er_curve(
  ermod_bin,
  output_type = "median_qi"
)

plot_er(ersim_med_qi, show_orig_data = TRUE) +
  xgxr::xgx_scale_x_log10()
```

plot_er_exp_sel *Plot exposure metric selection comparison*

Description

Plot ER curve for each exposure metric and compare them.

Usage

```
plot_er_exp_sel(x, n_draws_sim = NULL)
```

Arguments

x	An object of class `ermod_bin_exp_sel`
n_draws_sim	Number of draws to simulate response for each exposure value. Default is NULL (use all draws in the model object)

Value

No return value, called for plotting side effect.

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin_exp_sel <-
  dev_ermod_bin_exp_sel(
    data = d_sim_binom_cov_hgly2,
    var_resp = "AEFLAG",
    var_exp_candidates = c("AUCss_1000", "Cmaxss", "Cminss")
  )

plot_er_exp_sel(ermod_bin_exp_sel) + xgxr::xgx_scale_x_log10()
```

plot_er_gof *Default GOF plot for ER model*

Description

This is a wrapper function for `plot_er()` with default options for goodness-of-fit (GOF) plots for ER models.

Usage

```
plot_er_gof(
  x,
  add_boxplot = !is.null(var_group),
  boxplot_height = 0.15,
  show_boxplot_y_title = FALSE,
  var_group = NULL,
  n_bins = 4,
  qi_width_obs = 0.95,
  show_coef_exp = FALSE,
  coef_pos_x = NULL,
  coef_pos_y = NULL,
  coef_size = 4,
  qi_width_coef = 0.95,
  qi_width_sim = 0.95,
  show_caption = TRUE
)
```

Arguments

`x`	an object of class `ermod`, `ersim`,`ersim_med_qi`, or their subclasses
`add_boxplot`	Logical, whether to add a boxplot of exposure values. Default is `TRUE` if `var_group` is specified, otherwise `FALSE`.

`boxplot_height` Height of the boxplot relative to the main plot. Default is `0.15`.

`show_boxplot_y_title` Logical, whether to show the y-axis title for the boxplot. Default is `FALSE`.

`var_group` The column to use for grouping data for plotting. If specified, observed data points and boxplot will be grouped and colored by this column. Default is `NULL`.

`n_bins` Number of bins to use for observed probability summary. Only relevant for binary models. Default is `4`.

`qi_width_obs` Confidence level for the observed probability summary. Default is `0.95`.

`show_coef_exp` Logical, whether to show the credible interval of the exposure coefficient. Default is `FALSE`. This is only available for linear and linear logistic regression models.

`coef_pos_x` x-coordinate of the text label. If `NULL` (default), it is set to the minimum value for the exposure variable.

`coef_pos_y` y-coordinate of the text label. If `NULL` (default), it is set to 0.9 for logistic regression models and the maximum value of the response variable in the original data for linear regression models.

`coef_size` Size of the text label. Default is `4`.

`qi_width_coef` Width of the credible interval for the exposure coefficient. Default is `0.95`.

`qi_width_sim` Width of the quantile interval to summarize simulated draws. Default is `0.95`.

`show_caption` Logical, whether to show the caption note for the plot. Default is `TRUE`.

Details

The following code will generate the same plot:

```
plot_er(
  x,
  show_orig_data = TRUE,
  show_coef_exp = show_coef_exp,
  show_caption = show_caption,
  options_orig_data = list(
    add_boxplot = add_boxplot, boxplot_height = boxplot_height,
    show_boxplot_y_title = show_boxplot_y_title,
    var_group = var_group,
    n_bins = n_bins, qi_width = qi_width_obs
  ),
  options_coef_exp = list(
    qi_width = qi_width_coef, pos_x = coef_pos_x, pos_y = coef_pos_y,
    size = coef_size
  ),
  options_caption = list(
    orig_data_summary = TRUE, coef_exp = show_coef_exp
  ),
  qi_width_sim = qi_width_sim
)
```

Value

A ggplot object

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000"
)

plot_er_gof(ermod_bin, var_group = "Dose_mg", show_coef_exp = TRUE)
```

`print_coveff` *Format the covariate effect simulation results for printing*

Description

Format the covariate effect simulation results for printing

Usage

```
print_coveff(
  coveffsim,
  n_sigfig = 3,
  use_seps = TRUE,
  drop_trailing_dec_mark = TRUE
)
```

Arguments

`coveffsim` an object of class `coveffsim`

`n_sigfig` Number of significant figures to form value_label of continuous variables. See `gt::vec_fmt_number()` for details.

`use_seps` Whether to use separators for thousands in printing numbers. See `gt::vec_fmt_number()` for details.

`drop_trailing_dec_mark` Whether to drop the trailing decimal mark (".") in value_label of continuous variables. See `gt::vec_fmt_number()` for details.

Details

Note that n_sigfig, use_seps, and drop_trailing_dec_mark are only applied to the odds ratio and 95% CI columns; value_label column was already generated in an earlier step in build_spec_coveff() or sim_coveff().

Value

A data frame with the formatted covariate effect simulation results with the following columns:

- var_label: the label of the covariate
- value_label: the label of the covariate value
- value_annot: the annotation of the covariate value
- Odds ratio: the odds ratio of the covariate effect
- 95% CI: the 95% credible interval of the covariate effect

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

print_coveff(sim_coveff(ermod_bin))
```

p_direction *Probability of Direction (pd)*

Description

Compute the **Probability of Direction** (**pd**). Although differently expressed, this index is fairly similar (*i.e.*, is strongly correlated) to the frequentist **p-value**. See bayestestR::p_direction() and vignette("overview_of_vignettes", package = "bayestestR") > "Probability of Direction (pd)" page for details. For converting **pd** to a frequentist **p-value**, see bayestestR::pd_to_p().

Usage

```
p_direction(x, ...)

## S3 method for class 'ermod_bin'
p_direction(
  x,
  null = 0,
  as_p = FALSE,
  as_num = FALSE,
  direction = "two-sided",
  ...
)
```

Arguments

`x` An object of class `ermod_bin_*`

`...` Additional arguments passed to `bayestestR::p_direction()`.

`null` The null hypothesis value. Default is 0.

`as_p` If `TRUE`, the p-direction (pd) values are converted to a frequentist p-value using `bayestestR::pd_to_p()`. Only works when `as_num = TRUE`.

`as_num` If `TRUE`, the output is converted to a numeric value.

`direction` What type of p-value is requested or provided with as_p = TRUE. Can be `"two-sided"` (default, two tailed) or `"one-sided"` (one tailed).

Details

For the class `ermod_bin_*`, it only calculates the **pd** for the exposure variable.

Value

See `bayestestR::p_direction()` for details.

Examples

```
df_er_dr2 <-
  d_sim_binom_cov |>
  dplyr::filter(
    AETYPE == "dr2",
    ID %in% seq(1, 500, by = 5)
  ) |>
  dplyr::mutate(AUCss_1000 = AUCss / 1000, BHBA1C_5 = BHBA1C / 5)

ermod_bin <- dev_ermod_bin(
  data = df_er_dr2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5"
)

p_direction(ermod_bin, as_num = TRUE, as_p = TRUE)
```

run_kfold_cv *Run k-fold cross-validation*

Description

This function performs k-fold cross-validation using the appropriate model development function based on the class of the ermod object.

Usage

```
run_kfold_cv(ermod, newdata = NULL, k = 5, seed = NULL)
```

Arguments

ermod	An ermod object containing the model and data.
newdata	Optional new dataset to use instead of the original data. Default is NULL.
k	The number of folds for cross-validation. Default is 5.
seed	Random seed for reproducibility. Default is NULL.

Value

A kfold_cv_ermod class object containing the fitted models and holdout predictions for each fold.

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
  # Settings to make the example run faster
  chains = 2,
  iter = 1000
)

cv_results <- run_kfold_cv(ermod_bin, k = 3, seed = 123)

print(cv_results)
```

sim_coveff *Perform simulation of covariate effects for ER model*

Description

Perform simulation of covariate effects for ER model

Usage

```
sim_coveff(
  ermod,
  data = NULL,
  spec_coveff = NULL,
  output_type = "median_qi",
  qi_width = 0.9,
  qi_width_cov = 0.9
)
```

Arguments

ermod	an object of class ermod
data	an optional data frame to derive the covariate values for forest plots. If NULL (default), the data used to fit the model is used.
spec_coveff	you can supply spec_coveff to sim_coveff() or plot_coveff(), if you have already built it manually or with build_spec_coveff(). See build_spec_coveff() for detail.
output_type	Type of output. Currently only supports "median_qi" which returns the median and quantile interval.
qi_width	the width of the credible interval on the covariate effect. This translate to the width of the error bars in the forest plot.
qi_width_cov	the width of the quantile interval for continuous covariates in the forest plot. Default is 0.9 (i.e. visualize effect of covariate effect at their 5th and 95th percentile values).

Value

A data frame with class coveffsim containing the median and quantile interval of the covariate effects.

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

sim_coveff(ermod_bin)
```

sim_er *Simulate from ER model*

Description

Simulate from ER model

Usage

```
sim_er(
  ermod,
  newdata = NULL,
  n_draws_sim = NULL,
  seed_sample_draws = NULL,
  output_type = c("draws", "median_qi"),
  qi_width = 0.95,
  .nrow_cov_data = NULL
)
```

Arguments

`ermod` An object of class `ermod`

`newdata` New data to use for simulation. Default is NULL (use the data in the model object).

`n_draws_sim` Number of draws for simulation. If NULL (default), all draws in the model object are used.

`seed_sample_draws` Seed for sampling draws. Default is NULL.

`output_type` Type of output. "draws" returns the raw draws from the simulation, and "median_qi" returns the median and quantile interval.

`qi_width` Width of the quantile interval. Default is 0.95. Only used when `output_type = "median_qi"`.

`.nrow_cov_data` Number of rows in the covariate data, used for internal purposes. Users should not set this argument.

Value

ersim object, which is a tibble with the simulated responses with some additional information in object attributes. It has three types of predictions - .linpred, .epred, .prediction. .linpred and .epred are similar in a way that they both represent "expected response", i.e. without residual variability. They are the same for models with continuous endpoits (Emax model). For models with binary endpoints, .linpred is the linear predictor (i.e. on the logit scale) and .epred is on the probability scale. .prediction is the predicted response with residual variability (or in case of binary endpoint, the predicted yes (1) or no (0) for event occurrence). See tidybayes::add_epred_draws() for more details.

In case of output_type = "median_qi", it returns ersim_med_qi object.

See Also

calc_ersim_med_qi() for calculating median and quantile interval from ersim object (generated with output_type = "draws").

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

ersim <- sim_er(
  ermod_bin,
  n_draws_sim = 500, # This is set to make the example run faster
  output_type = "draws"
)

ersim_med_qi <- sim_er(
  ermod_bin,
  n_draws_sim = 500, # This is set to make the example run faster
  output_type = "median_qi"
)

ersim
ersim_med_qi
```

---

sim_er_new_exp *Simulate from ER model at specified exposure values*

Description

Simulate from ER model at specified exposure values

Usage

```
sim_er_new_exp(
  ermod,
  exposure_to_sim_vec = NULL,
  data_cov = NULL,
  n_draws_sim = NULL,
  seed_sample_draws = NULL,
  output_type = c("draws", "median_qi"),
  qi_width = 0.95
)

sim_er_curve(
  ermod,
  exposure_range = NULL,
  num_exposures = 51,
  data_cov = NULL,
  n_draws_sim = NULL,
  seed_sample_draws = NULL,
  output_type = c("draws", "median_qi"),
  qi_width = 0.95
)
```

Arguments

`ermod` An object of class `ermod`

`exposure_to_sim_vec` Vector of exposure values to simulate.

`data_cov` Data frame containing covariates to use for simulation, see details below.

`n_draws_sim` Number of draws for simulation. If NULL (default), all draws in the model object are used.

`seed_sample_draws` Seed for sampling draws. Default is NULL.

`output_type` Type of output. "draws" returns the raw draws from the simulation, and "median_qi" returns the median and quantile interval.

`qi_width` Width of the quantile interval. Default is 0.95. Only used when `output_type = "median_qi"`.

`exposure_range` Range of exposure values to simulate. If NULL (default), it is set to the range of the exposure variable in the original data for model development.

`num_exposures` Number of exposure values to simulate.

Details

Simulation dataset will be all combinations of covariates in data_cov and exposure values in exposure_to_sim_vec, so the run time can become very long if data_cov has many rows.

data_cov has to be supplied if ermod is a model with covariates. It is recommended that data_cov contains subject identifiers such as ID for post-processing.

Exposure values in data_cov will be ignored.

sim_er_curve() is a wrapper function for sim_er_new_exp() that use a range of exposure values to simulate the expected responses. Particularly useful for plotting the exposure-response curve.

Value

ersim object, which is a tibble with the simulated responses with some additional information in object attributes. It has three types of predictions - .linpred, .epred, .prediction. .linpred and .epred are similar in a way that they both represent "expected response", i.e. without residual variability. They are the same for models with continuous endpoits (Emax model). For models with binary endpoints, .linpred is the linear predictor (i.e. on the logit scale) and .epred is on the probability scale. .prediction is the predicted response with residual variability (or in case of binary endpoint, the predicted yes (1) or no (0) for event occurrence). See tidybayes::add_epred_draws() for more details.

In case of output_type = "median_qi", it returns ersim_med_qi object.

See Also

calc_ersim_med_qi() for calculating median and quantile interval from ersim object (generated with output_type = "draws").

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

ersim_new_exp_med_qi <- sim_er_new_exp(
  ermod_bin,
  exposure_to_sim_vec = seq(2, 6, by = 0.2),
  data_cov = dplyr::tibble(BHBA1C_5 = 4:10),
  n_draws_sim = 500, # This is set to make the example run faster
  output_type = "median_qi"
)

ersim_new_exp_med_qi
```

sim_er_new_exp_marg *Calculate marginal expected response for specified exposure values*

Description

Responses at specified exposure values are calculated for n_subj_sim subjects with different covariates (sampled from newdata), and the predicted responses are "marginalized" (averaged), resulting in marginal expected response on the population of interest.

Usage

```
sim_er_new_exp_marg(
  ermod,
  exposure_to_sim_vec = NULL,
  data_cov = extract_data(ermod),
  n_subj_sim = 100,
  n_draws_sim = 500,
  seed_sample_draws = NULL,
  output_type = c("draws", "median_qi"),
  qi_width = 0.95
)

sim_er_curve_marg(
  ermod,
  exposure_range = NULL,
  num_exposures = 51,
  data_cov = extract_data(ermod),
  n_subj_sim = 100,
  n_draws_sim = 500,
  seed_sample_draws = NULL,
  output_type = c("draws", "median_qi"),
  qi_width = 0.95
)
```

Arguments

ermod	An object of class ermod
exposure_to_sim_vec	Vector of exposure values to simulate.
data_cov	Data frame containing covariates to use for simulation. Different from sim_er_new_exp(), data_cov can be large as long as n_subj_sim is set to a reasonable number. Default is set to extract_data(ermod) which is the full data used to fit the model.
n_subj_sim	Maximum number of subjects to simulate. Default of 100 should be sufficient in many cases, as it's only used for marginal response calculation. Set to NULL to use all subjects in data_cov without resampling; in this case, be mindful of the computation time.

`n_draws_sim` Number of draws for simulation. Default is set to 500 to reduce computation time for marginal response calculation.

`seed_sample_draws`
Seed for sampling draws. Default is NULL.

`output_type` Type of output. "draws" returns the raw draws from the simulation, and "median_qi" returns the median and quantile interval.

`qi_width` Width of the quantile interval. Default is 0.95. Only used when `output_type = "median_qi"`.

`exposure_range` Range of exposure values to simulate. If NULL (default), it is set to the range of the exposure variable in the original data for model development.

`num_exposures` Number of exposure values to simulate.

Details

`sim_er_new_exp_marg()` returns a tibble with the marginal expected response for each exposure value in `exposure_to_sim_vec`.

`sim_er_curve_marg()` is a wrapper function for `sim_er_new_exp_marg()` that use a range of exposure values to simulate the marginal expected responses. Particularly useful for plotting the exposure-response curve.

Value

`ersim_marg` object, which is a tibble with the simulated marginal expected response with some additional information in object attributes. In case of `output_type = "median_qi"`, it returns `ersim_marg_med_qi` object.

See Also

`calc_ersim_med_qi()` for calculating median and quantile interval from `ersim_marg` object (generated with `output_type = "draws"`).

Examples

```
data(d_sim_binom_cov_hgly2)

ermod_bin <- dev_ermod_bin(
  data = d_sim_binom_cov_hgly2,
  var_resp = "AEFLAG",
  var_exposure = "AUCss_1000",
  var_cov = "BHBA1C_5",
)

ersim_new_exp_marg_med_qi <- sim_er_new_exp_marg(
  ermod_bin,
  exposure_to_sim_vec = seq(2, 6, by = 0.2),
  data_cov = dplyr::tibble(BHBA1C_5 = 4:10),
  n_subj_sim = NULL,
  n_draws_sim = 500, # This is set to make the example run faster
  output_type = "median_qi"
)

ersim_new_exp_marg_med_qi
```

Index

∗ **datasets**
d_sim_binom_cov, 15
d_sim_lin, 16

as_draws, 2
as_draws_array *(as_draws)*, 2
as_draws_df *(as_draws)*, 2
as_draws_list *(as_draws)*, 2
as_draws_matrix *(as_draws)*, 2
as_draws_rvars *(as_draws)*, 2

bayestestR::p_direction(), *34, 35*
bayestestR::pd_to_p(), *34, 35*
build_spec_coveff, 3
build_spec_coveff(), *4*, *18*, *25*, *34*, *37*
build_spec_coveff_one_variable *(edit_spec_coveff)*, 17
build_spec_coveff_one_variable(), *17, 18*

calc_ersim_med_qi, 5
calc_ersim_med_qi(), *39*, *41*, *43*
coef.ermod *(ermod_method)*, 20

d_sim_binom_cov, 15
d_sim_binom_cov_hgly2 *(d_sim_binom_cov)*, 15
d_sim_lin, 16
dev_ermod_bin, 6
dev_ermod_bin(), *6*
dev_ermod_bin_cov_sel, 7
dev_ermod_bin_cov_sel(), *4*
dev_ermod_bin_emax *(dev_ermod_emax)*, 11
dev_ermod_bin_emax_exp_sel *(dev_ermod_emax_exp_sel)*, 13
dev_ermod_bin_exp_sel, 9
dev_ermod_emax, 11
dev_ermod_emax_exp_sel, 13
dev_ermod_lin *(dev_ermod_bin)*, 6
dev_ermod_lin(), *6*
dev_ermod_lin_cov_sel *(dev_ermod_bin_cov_sel)*, 7
dev_ermod_lin_exp_sel *(dev_ermod_bin_exp_sel)*, 9

edit_spec_coveff, 17
ermod_cov_sel_method, 19
ermod_exp_sel_method, 19
ermod_method, 20
ersim_method, 21
eval_ermod, 21
extract_coef_exp_ci, 23
extract_data *(extract_method)*, 23
extract_data(), *24*
extract_ermod, *23*
extract_ersim, *23*
extract_exp_sel_comp *(extract_method)*, 23
extract_exp_sel_comp(), *24*
extract_exp_sel_list_model *(extract_method)*, 23
extract_exp_sel_list_model(), *24*
extract_method, 23
extract_mod *(extract_method)*, 23
extract_mod(), *24*
extract_var_cov *(extract_method)*, 23
extract_var_cov(), *24*
extract_var_exposure *(extract_method)*, 23
extract_var_exposure(), *24*
extract_var_resp *(extract_method)*, 23
extract_var_resp(), *24*
extract_var_selected *(extract_method)*, 23
extract_var_selected(), *24*

gt::vec_fmt_number(), *4*, *18*, *33*

loo, 24
loo::loo(), *24*

p_direction, 34
plot.ermod_bin (ermod_method), 20
plot.ermod_cov_sel (ermod_cov_sel_method), 19
plot.ermod_exp_sel (ermod_exp_sel_method), 19
plot.ersim (ersim_method), 21
plot.ersim_med_qi (ersim_method), 21
plot_cov_sel, 26
plot_coveff, 25
plot_coveff(), *4*, *25*, *37*
plot_er, 27
plot_er(), *31*
plot_er_exp_sel, 30
plot_er_gof, 31
plot_submod_performance (plot_cov_sel), 26
plot_submod_performance(), *26*
plot_var_ranking (plot_cov_sel), 26
plot_var_ranking(), *26*
posterior::as_draws(), *2*
print.ermod (ermod_method), 20
print.ermod_cov_sel (ermod_cov_sel_method), 19
print.ermod_exp_sel (ermod_exp_sel_method), 19
print_coveff, 33

replace_spec_coveff (edit_spec_coveff), 17
replace_spec_coveff(), *17*, *18*
rstan::sampling(), *12*, *14*
rstanemax::stan_emax(), *12*, *14*
rstanemax::stan_emax_binary(), *12*, *14*
run_kfold_cv, 36

sim_coveff, 37
sim_coveff(), *4*, *25*, *34*, *37*
sim_er, 38
sim_er_curve (sim_er_new_exp), 39
sim_er_curve(), *41*
sim_er_curve_marg (sim_er_new_exp_marg), 42
sim_er_curve_marg(), *43*
sim_er_new_exp, 39
sim_er_new_exp(), *41*, *42*
sim_er_new_exp_marg, 42
sim_er_new_exp_marg(), *43*
summary.ermod (ermod_method), 20

tidybayes::add_epred_draws(), *39*, *41*