

Package ‘bregr’

August 1, 2026

Title Easy and Efficient Batch Processing of Regression Models

Version 1.5.0

Maintainer Shixiang Wang <w_shixiang@163.com>

Description Easily processes batches of univariate or multivariate regression models. Returns results in a tidy format and generates visualization plots for straightforward interpretation (Wang, Shixiang, et al. (2025) <[DOI:10.1002/mdr.2.70028](https://doi.org/10.1002/mdr.2.70028)>).

License GPL (>= 3)

URL <https://github.com/WangLabCSU/bregr>,
<https://wanglabcsu.github.io/bregr/>

BugReports <https://github.com/WangLabCSU/bregr/issues>

Depends R (>= 4.1.0)

Imports broom, broom.helpers, cli, dplyr, forestploter, ggplot2, glue, insight, lifecycle, mirai, purrr, rlang (>= 1.1.0), S7, survival, tibble, utils, vctrs (>= 0.5.0)

Suggests broom.mixed, fs, ggalign, ggnewscale, ggstats, ggstatsplot, gtsummary, ids, knitr, lme4, merDeriv, parallel, qs2, rmarkdown, testthat (>= 3.0.0), clusterProfiler, glmnet, UCSCXenaShiny, visreg

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Shixiang Wang [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-9855-7357>>),
Yun Peng [aut] (ORCID: <<https://orcid.org/0000-0003-2801-3332>>),
Chenyang Shu [aut]

Repository CRAN

Date/Publication 2026-08-01 05:00:02 UTC

Contents

accessors	2
avails	4
breg	5
br_compare_models	6
br_diagnose	8
br_get_model_stats	9
br_predict	10
br_seed	11
br_seed_model	13
br_seed_screen	15
br_seed_select	16
br_show_coxph_diagnostics	18
br_show_fitted_line	20
br_show_fitted_line_2d	21
br_show_forest	22
br_show_forest_circle	23
br_show_forest_comparison	25
br_show_forest_ggstats	26
br_show_forest_ggstatsplot	27
br_show_nomogram	28
br_show_residuals	29
br_show_risk_network	31
br_show_survival_curves	32
br_show_table	33
br_show_table_gt	34
pipeline	35
polar_connect	39
polar_init	40
print.breg	41
print.breg_comparison	41
Index	43

accessors

Accessor functions for breg objects

Description

[Stable]

These functions provide access to components of breg objects, serving as counterparts to the `br_set_*`() functions. Some functions include additional arguments for extended functionality.

Usage

```

br_get_data(obj)

br_get_y(obj)

br_get_x(obj)

br_get_n_x(obj)

br_get_x2(obj)

br_get_n_x2(obj)

br_get_group_by(obj)

br_get_config(obj)

br_get_models(obj, idx = NULL, auto_drop = TRUE)

br_get_model(obj, idx)

br_get_model_names(obj)

br_rename_models(obj, new_names)

br_get_results(obj, tidy = FALSE, ...)

```

Arguments

<code>obj</code>	A breg object.
<code>idx</code>	Index or names (focal variables) of the model(s) to return. Default returns all.
<code>auto_drop</code>	If TRUE, automatically drop the list if only one model is selected.
<code>new_names</code>	Character vector to replace existing model names.
<code>tidy</code>	If TRUE return tidy (compact) results, otherwise return comprehensive results. The tidy results are obtained from <code>broom::tidy()</code> while comprehensive results are obtained from <code>broom.helpers::tidy_plus_plus()</code> . The results can be configured when run with <code>br_run()</code> .
<code>...</code>	Subset operations passing to <code>dplyr::filter()</code> to filter results.

Value

Output depends on the function called:

- `br_get_data()` returns a `data.frame`.
- `br_get_y()`, `br_get_x()`, `br_get_x2()` return modeling terms.
- `br_get_n_x()` and `br_get_n_x2()` return the length of terms `x` and `x2`.

- `br_get_group_by()` returns variable(s) for group analysis.
- `br_get_config()` returns modeling method and extra arguments.
- `br_get_models()` returns all or a subset of constructed models.
- `br_get_model()` returns a subset of constructed models.
- `br_get_model_names()` returns all model names.
- `br_rename_models()` returns a renamed object.
- `br_get_results()` returns modeling result data.frame.

See Also

[pipeline](#) for building breg objects.

Other accessors: [br_diagnose\(\)](#), [br_get_model_stats\(\)](#), [br_predict\(\)](#)

Examples

```
m <- br_pipeline(mtcars,
  y = "mpg",
  x = colnames(mtcars)[2:4],
  x2 = "vs",
  method = "gaussian"
)
br_get_data(m)
br_get_y(m)
br_get_x(m)
br_get_n_x(m)
br_get_x2(m)
br_get_n_x2(m)
br_get_group_by(m)
br_get_config(m)
br_get_models(m)
br_get_models(m, 1)
br_get_n_x2(m)
br_get_results(m)
br_get_results(m, tidy = TRUE)
br_get_results(m, tidy = TRUE, term == "cyl")
```

avails

Package availability

Description

[Stable]

Package resource, definitions ready for use.

Usage

```
br_avail_methods()

br_avail_methods_use_exp()

br_avail_method_config(method)
```

Arguments

method Method for model construction. See [br_avail_methods\(\)](#) for available options.

Value

A character vector representing the available methods or options.

Functions

- `br_avail_methods()`: Returns available modeling methods. This correlates to [br_set_model\(\)](#).
- `br_avail_methods_use_exp()`: Returns available modeling methods which set `exponentiate=TRUE` at default by **breg**.
- `br_avail_method_config()`: Returns model configs for specified method to generate modeling templates.

See Also

[pipeline](#) for building breg objects.

breg	<i>Creates a new breg-class object</i>
------	--

Description

[Stable]

Constructs a breg-class object containing regression model specifications and results.

Usage

```
breg(
  data = NULL,
  y = NULL,
  x = NULL,
  x2 = NULL,
  group_by = NULL,
  config = NULL,
  models = list(),
```

```

    results = NULL,
    results_tidy = NULL
  )

```

Arguments

data	A <code>data.frame</code> containing modeling data.
y	Character vector of dependent variable names.
x	Character vector of focal independent variable names.
x2	Optional character vector of control variable names.
group_by	Optional character vector specifying grouping column.
config	List of model configuration parameters.
models	List containing fitted model objects.
results	A <code>data.frame</code> of model results from <code>broom.helpers::tidy_plus_plus()</code> .
results_tidy	A <code>data.frame</code> of tidy model results from <code>broom::tidy()</code> .

Value

A constructed `breg` object.

Examples

```

obj <- breg()
obj
print(obj, raw = TRUE)

```

br_compare_models	<i>Compare univariate and multivariate models</i>
-------------------	---

Description

[Experimental]

This function builds both univariate models (each predictor separately) and a multivariate model (all predictors together), then combines the results for comparison. This is useful for understanding how predictor effects change when accounting for other variables.

Usage

```

br_compare_models(
  data,
  y,
  x,
  x2 = NULL,
  method,

```

```

    ...,
    n_workers = 1L,
    model_args = list(),
    run_args = list()
  )

```

Arguments

<code>data</code>	A <code>data.frame</code> containing all necessary variables for analysis.
<code>y</code>	Character vector specifying dependent variables (response variables). For GLM models, this is typically a single character (e.g., "outcome"). For Cox-PH models, it should be a length-2 vector in the format <code>c("time", "status")</code> .
<code>x</code>	Character vector specifying focal independent terms (predictors). These will be modeled both individually (univariate) and together (multivariate).
<code>x2</code>	Character vector specifying control independent terms (predictors, optional). These are included in all models (both univariate and multivariate).
<code>method</code>	Method for model construction. See <code>br_set_model()</code> for details.
<code>...</code>	Additional arguments passed to <code>br_run()</code> .
<code>n_workers</code>	Integer, indicating number of workers for parallel processing.
<code>model_args</code>	A list of arguments passed to <code>br_set_model()</code> .
<code>run_args</code>	A list of arguments passed to <code>br_run()</code> .

Value

A list with class `breg_comparison` containing:

- `univariate`: breg object with univariate model results
- `multivariate`: breg object with multivariate model results
- `combined_results`: Combined results data frame with a mode column
- `combined_results_tidy`: Combined tidy results with a mode column

See Also

Other `br_compare`: `br_show_forest_comparison()`

Examples

```

# Compare univariate vs multivariate for Cox models
lung <- survival::lung |>
  dplyr::filter(ph.ecog != 3)
lung$ph.ecog <- factor(lung$ph.ecog)

comparison <- br_compare_models(
  lung,
  y = c("time", "status"),
  x = c("ph.ecog", "ph.karno", "pat.karno"),
  x2 = c("age", "sex"),

```

```

    method = "coxph"
  )

  # View combined results
  comparison$combined_results_tidy

  # Create forest plot comparison
  br_show_forest_comparison(comparison)

```

br_diagnose

*Diagnose regression models***Description****[Experimental]**

Universal diagnostic function that performs appropriate diagnostics based on the model type. For Cox models, tests proportional hazards assumption using Schoenfeld residuals and provides comprehensive Cox diagnostics. For other models, provides general diagnostic information.

Usage

```
br_diagnose(breg, idx = NULL, transform = "km", ...)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Index or name (focal variable) of the model(s) to diagnose. If NULL, diagnoses all models in the breg object.
transform	Character string specifying how to transform time for Cox PH tests. Options are "km" (Kaplan-Meier), "rank", "identity", or a function.
...	Additional arguments passed to specific diagnostic functions.

Value

A list containing diagnostic results for each model.

See Also

Other accessors: [accessors](#), [br_get_model_stats\(\)](#), [br_predict\(\)](#)

Examples

```

# Create models
mds <- br_pipeline(
  survival::lung,
  y = c("time", "status"),
  x = colnames(survival::lung)[6:10],
  x2 = c("age", "sex"),

```

```

    method = "coxph"
  )

  # Diagnose models (includes PH testing for Cox models)
  diagnostics <- br_diagnose(mds)
  print(diagnostics)

```

br_get_model_stats	<i>Get model-level statistics in tidy format</i>
--------------------	--

Description

[Experimental]

Extracts per-model summary statistics (N, events, C-index, AIC, PH test, etc.) and returns them as a tidy data.frame. This complements [br_get_results\(\)](#) which provides per-term estimates.

Usage

```
br_get_model_stats(breg, idx = NULL)
```

Arguments

breg	A regression object with results.
idx	Index or name (focal variable) of the model(s). If NULL, returns stats for all models.

Value

A data.frame with one row per model. Columns depend on model type:

- Cox models: model, n, events, c_index, aic, ph_test_p, lr_test_p
- GLM models: model, n, aic, deviance, df_residual
- LM models: model, n, r_squared, adj_r_squared, df_residual

See Also

Other accessors: [accessors](#), [br_diagnose\(\)](#), [br_predict\(\)](#)

Examples

```

m <- br_pipeline(survival::lung,
  y = c("time", "status"),
  x = colnames(survival::lung)[6:10],
  method = "coxph"
)
br_get_model_stats(m)

```

br_predict

*Predict method for breg objects***Description****[Experimental]**

Generate predictions from fitted models in a breg object. For Cox regression models, returns linear predictors (log relative hazard). For other models, returns predicted values.

Usage

```
br_predict(obj, newdata = NULL, idx = NULL, type = NULL)
```

Arguments

obj	A breg object with fitted models.
newdata	Optional data frame for predictions. If NULL, uses original data.
idx	Model index, an integer or string.
type	Type of prediction. For Cox models: "lp" (linear predictor, default) or "risk" (relative risk). For other models: "response" (default) or "link".

Value

Typically, a numeric vector of predictions.

See Also

Other accessors: [accessors](#), [br_diagnose\(\)](#), [br_get_model_stats\(\)](#)

Examples

```
# Cox regression example
if (requireNamespace("survival", quietly = TRUE)) {
  lung <- survival::lung |> dplyr::filter(ph.ecog != 3)
  mds <- br_pipeline(
    lung,
    y = c("time", "status"),
    x = c("age", "ph.ecog"),
    x2 = "sex",
    method = "coxph"
  )
  scores <- br_predict(mds)
  head(scores)
}
```

br_seed

*Run the full SEED gene selection pipeline***Description****[Experimental]**

Implements the complete SEED (Selection of Essential prognostic genes from Expression Data) pipeline for identifying cancer prognosis target genes from RNA-seq data without control samples.

The pipeline runs four steps sequentially:

1. **Preliminary screening:** Univariate regression (Cox/logistic/Spearman) per gene against clinical indicators
2. **Enrichment analysis** (optional): Pathway enrichment to refine gene set
3. **Gene selection:** Lasso -> multivariate -> stepwise regression
4. **Model development:** Build risk score from final gene set

Based on Yang, H. et al. (2025) *Cancer Letters*, DOI: 10.1016/j.canlet.2025.217960.

Usage

```
br_seed(
  data,
  y,
  y_type,
  genes = NULL,
  x2 = NULL,
  p_threshold = 0.05,
  n_workers = 1L,
  enrich = NULL,
  do_select = TRUE,
  lasso_lambda = c("lambda.min", "lambda.1se"),
  lasso_nfolds = 10,
  step_direction = c("both", "backward", "forward"),
  step_k = 2,
  risk_scale = c("none", "zscore"),
  seed = NULL,
  ...
)
```

Arguments

data	A data.frame containing clinical variables and gene expression columns side by side. Gene columns must be numeric.
y	Character vector specifying clinical indicators. For survival: a length-2 vector c("time", "status"). For binary/ordinal: a single column name. When screening against multiple indicators, pass a named list, e.g., list(OS = c("time", "status"), ORR = "response").

y_type	Character vector of indicator types: "survival", "binary", or "ordinal". Must match y in length.
genes	Character vector of gene column names. If NULL, all numeric columns not in y or x2 are used.
x2	Optional character vector of adjustment covariates included in all models.
p_threshold	Numeric significance threshold for gene filtering (0 to 1). Default is 0.05.
n_workers	Integer number of parallel workers for batch regression. Passed to br_pipeline() .
enrich	Optional enrichment method. One of: • NULL (default): skip enrichment • A string naming an org.*.eg.db package (e.g., "org.Hs.eg.db") • A function taking a character vector of gene symbols and returning a character vector of enriched genes
do_select	Logical. If TRUE (default), run gene selection step.
lasso_lambda	Lambda selection criterion: "lambda.min" (default) or "lambda.1se".
lasso_nfolds	Number of folds for cross-validation in Lasso. Default 10.
step_direction	Direction for stepwise selection: "both" (default), "backward", or "forward".
step_k	The penalty multiplier for AIC in stepwise selection. Default 2 (standard AIC).
risk_scale	How to scale expression values before computing risk score. "none" (default) uses raw expression; "zscore" centers and scales.
seed	Integer seed for reproducibility of Lasso cross-validation.
...	Additional arguments passed to br_pipeline() .

Value

A list with class `breg_seed` containing:

- screen: `breg_seed_screen` object from Step 1
- select: `breg_seed_select` object from Step 3 (if `do_select = TRUE`)
- model: `breg_seed_model` object from Step 4
- risk_score: numeric vector of risk scores per sample

See Also

Other `br_seed`: [br_seed_model\(\)](#), [br_seed_screen\(\)](#), [br_seed_select\(\)](#)

Examples

```
set.seed(123)
n <- 100
test_data <- data.frame(
  time = rexp(n, 0.1),
  status = sample(0:1, n, replace = TRUE),
  GENE1 = rnorm(n),
  GENE2 = rnorm(n, mean = 0.8 * (1:n) / n),
  GENE3 = rnorm(n, mean = -0.5 * (1:n) / n),
```

```

    GENE4 = rnorm(n),
    GENE5 = rnorm(n),
    age = rnorm(n, 60, 10)
  )

  if (rlang::is_installed("glmnet")) {
    res <- br_seed(test_data,
      y = c("time", "status"),
      y_type = "survival",
      genes = c("GENE1", "GENE2", "GENE3", "GENE4", "GENE5"),
      x2 = "age",
      p_threshold = 0.3,
      seed = 42
    )
    print(res)
  }

```

br_seed_model

*Build a risk score model from selected genes***Description****[Experimental]**

Step 4 of the SEED pipeline: builds a final regression model from the selected gene set and computes a risk score for each sample. The risk score is the linear predictor (gene expression x coefficient) summed across genes.

Usage

```

br_seed_model(
  data,
  y,
  y_type,
  genes,
  x2 = NULL,
  risk_scale = c("none", "zscore")
)

```

Arguments

data	A data.frame containing clinical variables and gene expression columns side by side. Gene columns must be numeric.
y	Character vector specifying clinical indicators. For survival: a length-2 vector c("time", "status"). For binary/ordinal: a single column name. When screening against multiple indicators, pass a named list, e.g., list(OS = c("time", "status"), ORR = "response").

y_type	Character: "survival" or "binary". Ordinal indicators are not supported for selection/modeling steps.
genes	Character vector of final gene names, or a breg_seed_select object from which final_genes are extracted.
x2	Optional character vector of adjustment covariates included in all models.
risk_scale	How to scale expression values before computing risk score. "none" (default) uses raw expression; "zscore" centers and scales.

Value

A list with class `breg_seed_model` containing:

- `model`: breg object with the final model
- `coefficients`: data.frame of gene coefficients
- `risk_score`: numeric vector, one per sample
- `risk_score_formula`: character string representation of the risk formula

See Also

Other `br_seed`: [br_seed\(\)](#), [br_seed_screen\(\)](#), [br_seed_select\(\)](#)

Examples

```
set.seed(123)
n <- 100
test_data <- data.frame(
  time = rexp(n, 0.1),
  status = sample(0:1, n, replace = TRUE),
  GENE1 = rnorm(n),
  GENE2 = rnorm(n, mean = 0.8 * (1:n) / n),
  GENE3 = rnorm(n, mean = -0.5 * (1:n) / n),
  age = rnorm(n, 60, 10)
)

mod <- br_seed_model(test_data,
  y = c("time", "status"),
  y_type = "survival",
  genes = c("GENE1", "GENE2", "GENE3"),
  x2 = "age"
)
print(mod)
```

br_seed_screen	<i>Screen genes via univariate regression</i>
----------------	---

Description

[Experimental]

Step 1 of the SEED pipeline: runs univariate regression for each gene against one or more clinical indicators. Supports Cox regression (survival), logistic regression (binary), and Spearman correlation (ordinal). When multiple indicators are provided, genes significant across *all* indicators are retained (intersection).

Usage

```
br_seed_screen(
  data,
  y,
  y_type,
  genes = NULL,
  x2 = NULL,
  p_threshold = 0.05,
  n_workers = 1L,
  ...
)
```

Arguments

data	A <code>data.frame</code> containing clinical variables and gene expression columns side by side. Gene columns must be numeric.
y	Character vector specifying clinical indicators. For survival: a length-2 vector <code>c("time", "status")</code> . For binary/ordinal: a single column name. When screening against multiple indicators, pass a named list, e.g., <code>list(OS = c("time", "status"), ORR = "response")</code> .
y_type	Character vector of indicator types: "survival", "binary", or "ordinal". Must match y in length.
genes	Character vector of gene column names. If <code>NULL</code> , all numeric columns not in y or x2 are used.
x2	Optional character vector of adjustment covariates included in all models.
p_threshold	Numeric significance threshold for gene filtering (0 to 1). Default is 0.05.
n_workers	Integer number of parallel workers for batch regression. Passed to <code>br_pipeline()</code> .
...	Additional arguments passed to <code>br_pipeline()</code> .

Value

A list with class `breg_seed_screen` containing:

- `results`: data.frame of screening results (gene, indicator, p.value, etc.)
- `genes`: character vector of significant genes (intersection across indicators)
- `breg`: the underlying breg object (for regression-based screening)
- `n_genes_in`, `n_genes_out`: counts for progress reporting

See Also

Other `br_seed`: [br_seed\(\)](#), [br_seed_model\(\)](#), [br_seed_select\(\)](#)

Examples

```
set.seed(123)
n <- 100
# Create gene with real survival signal
test_data <- data.frame(
  time = rexp(n, 0.1),
  status = sample(0:1, n, replace = TRUE),
  GENE1 = rnorm(n),
  GENE2 = rnorm(n, mean = 0.8 * (1:n) / n),
  GENE3 = rnorm(n),
  age = rnorm(n, 60, 10)
)

# Single survival indicator; use relaxed p_threshold for example data
res <- br_seed_screen(test_data,
  y = c("time", "status"),
  y_type = "survival",
  genes = c("GENE1", "GENE2", "GENE3"),
  x2 = "age",
  p_threshold = 0.5
)
print(res)
```

`br_seed_select`
Select genes via Lasso -> multivariate -> stepwise regression

Description**[Experimental]**

Step 3 of the SEED pipeline: refines the gene set through three sequential stages — Lasso regularization, multivariate regression, and stepwise selection. The final gene set is the intersection of genes retained by all three methods.

Usage

```
br_seed_select(
  data,
  y,
  y_type,
  genes,
  x2 = NULL,
  p_threshold = 0.05,
  lasso_lambda = c("lambda.min", "lambda.1se"),
  lasso_nfolds = 10,
  step_direction = c("both", "backward", "forward"),
  step_k = 2,
  seed = NULL
)
```

Arguments

data	A data.frame containing clinical variables and gene expression columns side by side. Gene columns must be numeric.
y	Character vector specifying clinical indicators. For survival: a length-2 vector c("time", "status"). For binary/ordinal: a single column name. When screening against multiple indicators, pass a named list, e.g., list(OS = c("time", "status"), ORR = "response").
y_type	Character: "survival" or "binary". Ordinal indicators are not supported for selection/modeling steps.
genes	Character vector of gene names (typically from br_seed_screen()), or a breg_seed_screen object from which genes are extracted automatically.
x2	Optional character vector of adjustment covariates included in all models.
p_threshold	Numeric significance threshold for gene filtering (0 to 1). Default is 0.05.
lasso_lambda	Lambda selection criterion: "lambda.min" (default) or "lambda.1se".
lasso_nfolds	Number of folds for cross-validation in Lasso. Default 10.
step_direction	Direction for stepwise selection: "both" (default), "backward", or "forward".
step_k	The penalty multiplier for AIC in stepwise selection. Default 2 (standard AIC).
seed	Integer seed for reproducibility of Lasso cross-validation.

Value

A list with class `breg_seed_select` containing:

- `lasso_genes`, `multivariate_genes`, `stepwise_genes`: gene sets from each stage
- `final_genes`: intersection of all three gene sets
- `lasso_model`, `multivariate_model`, `stepwise_model`: fitted model objects

See Also

Other `br_seed`: [br_seed\(\)](#), [br_seed_model\(\)](#), [br_seed_screen\(\)](#)

Examples

```
set.seed(123)
n <- 100
test_data <- data.frame(
  time = rexp(n, 0.1),
  status = sample(0:1, n, replace = TRUE),
  GENE1 = rnorm(n),
  GENE2 = rnorm(n, mean = 0.8 * (1:n) / n),
  GENE3 = rnorm(n, mean = -0.5 * (1:n) / n),
  GENE4 = rnorm(n),
  GENE5 = rnorm(n),
  age = rnorm(n, 60, 10)
)

if (rlang::is_installed("glmnet")) {
  sel <- br_seed_select(test_data,
    y = c("time", "status"),
    y_type = "survival",
    genes = c("GENE1", "GENE2", "GENE3", "GENE4", "GENE5"),
    x2 = "age",
    seed = 42
  )
  print(sel)
}
```

br_show_coxph_diagnostics

Show Cox proportional hazards model diagnostic plots

Description

[Experimental]

Creates diagnostic plots specifically for Cox regression models. Focuses on Schoenfeld residuals plots to assess proportional hazards assumption and other Cox-specific diagnostics. Inspired by [survminer::ggcoxzph](#) with enhanced visualization and computation optimizations to work in **breg**.

Usage

```
br_show_coxph_diagnostics(
  breg,
  idx = 1,
  type = "schoenfeld",
  resid = TRUE,
  se = TRUE,
  point_col = "red",
  point_size = 1,
  point_alpha = 0.6,
```

```
    ...
  )
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Index or name (focal variable) of the Cox model to plot. Must be a single model.
type	Type of Cox diagnostic plot. Options: "schoenfeld" (default for Schoenfeld residuals), "martingale" (martingale residuals), "deviance" (deviance residuals).
resid	Logical, if TRUE the residuals are included on the plot along with smooth fit.
se	Logical, if TRUE confidence bands at two standard errors will be added.
point_col	Color for residual points.
point_size	Size for residual points.
point_alpha	Alpha (transparency) for residual points.
...	Additional arguments passed to <code>survival::cox.zph</code> .

Value

A ggplot2 object or list of plots.

See Also

Other `br_show`: `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`, `br_show_table_gt()`

Examples

```
# Create Cox models
mds <- br_pipeline(
  survival::lung,
  y = c("time", "status"),
  x = colnames(survival::lung)[6:10],
  x2 = c("age", "sex"),
  method = "coxph"
)

# Show Cox diagnostic plots
p1 <- br_show_coxph_diagnostics(mds, idx = 1)
p1
p2 <- br_show_coxph_diagnostics(mds, type = "martingale")
p2
```

br_show_fitted_line *Show fitted regression line with visreg interface*

Description

[Stable]

Provides an interface to visualize the model results with **visreg** package, to show how a predictor variable x affects an outcome y .

Usage

```
br_show_fitted_line(breg, idx = 1, ...)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Length-1 vector. Index or name (focal variable) of the model. This is different from <code>idx</code> in <code>br_show_forest_ggstats</code> , only one model is supported to visualized here, so only length-1 vector is supported as <code>idx</code> .
...	Arguments passing to <code>visreg::visreg()</code> excepts <code>fit</code> and <code>data</code> . See <code>visreg::visreg()</code> for available arguments.

Value

A plot

Note

Starting with **visreg** 3.0, all plots use `ggplot2` (the `gg` argument has been removed) and several arguments have been renamed to use `snake_case`: `line.par` -> `line`, `fill.par` -> `fill`, `points.par` -> `points`, `band.par` -> `band`. See `vignette("migrating-to-3-0", package = "visreg")` for full details.

See Also

Other `br_show`: `br_show_coxph_diagnostics()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`, `br_show_table_gt()`

Examples

```
if (rlang::is_installed("visreg")) {
  m <- br_pipeline(mtcars,
    y = "mpg",
    x = colnames(mtcars)[2:4],
    x2 = "vs",
```

```

    method = "gaussian"
  )

  if (interactive()) {
    br_show_fitted_line(m)
  }
  br_show_fitted_line(m, xvar = "cyl")
}

```

br_show_fitted_line_2d

Show 2d fitted regression line with visreg interface

Description

[Stable]

Similar to [br_show_fitted_line\(\)](#), but visualize how *two variables* interact to affect the response in regression models.

Usage

```
br_show_fitted_line_2d(breg, idx = 1, ...)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Length-1 vector. Index or name (focal variable) of the model. This is different from <code>idx</code> in br_show_forest_ggstats , only one model is supported to visualized here, so only length-1 vector is supported as <code>idx</code> .
...	Arguments passing to visreg::visreg2d() excepts <code>fit</code> and <code>data</code> . See visreg::visreg2d() for available arguments.

Value

A plot

Note

Starting with **visreg** 3.0, all plots use `ggplot2` (the `gg` argument has been removed) and several arguments have been renamed to use `snake_case`. See the note in [br_show_fitted_line\(\)](#) for details.

See Also

Other `br_show`: [br_show_coxph_diagnostics\(\)](#), [br_show_fitted_line\(\)](#), [br_show_forest\(\)](#), [br_show_forest_circle\(\)](#), [br_show_forest_ggstats\(\)](#), [br_show_forest_ggstatsplot\(\)](#), [br_show_nomogram\(\)](#), [br_show_residuals\(\)](#), [br_show_risk_network\(\)](#), [br_show_survival_curves\(\)](#), [br_show_table\(\)](#), [br_show_table_gt\(\)](#)

Examples

```
if (rlang::is_installed("visreg")) {
  m <- br_pipeline(mtcars,
    y = "mpg",
    x = colnames(mtcars)[2:4],
    x2 = "vs",
    method = "gaussian"
  )

  br_show_fitted_line_2d(m, xvar = "cyl", yvar = "mpg")
}
```

br_show_forest

Show a forest plot for regression results

Description

[Stable]

This function takes regression results and formats them into a forest plot display. It handles:

- Formatting of estimates, CIs and p-values
- Automatic x-axis limits calculation
- Cleaning of redundant group/focal variable labels
- Custom subsetting and column dropping The function uses `forestploter::forest()` internally for the actual plotting.

Usage

```
br_show_forest(
  breg,
  clean = TRUE,
  rm_controls = FALSE,
  ...,
  subset = NULL,
  drop = NULL,
  tab_headers = NULL,
  log_first = FALSE
)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
clean	Logical indicating whether to clean/condense redundant group/focal variable labels. If TRUE, remove "Group" or "Focal" variable column when the values in the result table are the same (before performing subset and drop), and reduce repeat values in column "Group", "Focal", and "Variable".

rm_controls	If TRUE, remove control terms.
...	Additional arguments passed to <code>forestploter::forest()</code> , run <code>vignette("forestploter-post", "forestploter")</code> to see more plot options. For example, use <code>ticks_at</code> to specify custom ticks, generally a vector of 4-5 elements.
subset	Expression for subsetting the results data (<code>br_get_results(breg)</code>).
drop	Column indices to drop from the display table.
tab_headers	Character vector of custom column headers (must match number of displayed columns).
log_first	Log transformed the estimates and their confident intervals. For only log scaled axis of the forest, use <code>x_trans = "log"</code> .

Value

A plot

See Also

Other `br_show`: `br_show_coxph_diagnostics()`, `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`, `br_show_table_gt()`

Examples

```
m <- br_pipeline(mtcars,
  y = "mpg",
  x = colnames(mtcars)[2:4],
  x2 = "vs",
  method = "gaussian"
)
br_show_forest(m)
br_show_forest(m, clean = TRUE, drop = 3)
br_show_forest(m, clean = FALSE)
```

`br_show_forest_circle` *Show a circular forest plot for regression results*

Description**[Experimental]**

This function creates a circular (polar) forest plot from regression results, providing an alternative visualization to the traditional linear forest plot. The function uses the same input as `br_show_forest()` but displays the results in a circular format using `ggplot2::coord_polar()`.

Usage

```
br_show_forest_circle(
  breg,
  rm_controls = FALSE,
  style = c("points", "bars"),
  ref_line = TRUE,
  sort_by = c("none", "estimate", "estimate_desc", "pvalue", "variable"),
  subset = NULL,
  log_first = FALSE
)
```

Arguments

breg	A regression object with results.
rm_controls	If TRUE, remove control terms.
style	Character string specifying the style of circular forest plot. Options are: "points" (default): Display point estimates with error bars in circular format "bars": Display as bars with points overlaid
ref_line	Logical or numeric. If TRUE, shows reference circle at default value (1 for exponentiated estimates, 0 for regular estimates). If numeric, shows reference circle at specified value. If FALSE, no reference circle is shown.
sort_by	Character string specifying how to sort the variables. Options are: "none" (default): No sorting, use original order "estimate": Sort by effect estimate (ascending) "estimate_desc": Sort by effect estimate (descending) "pvalue": Sort by p-value (ascending, most significant first) "variable": Sort alphabetically by variable name
subset	Expression for subsetting the results data (br_get_results(breg)).
log_first	Log transformed the estimates and their confident intervals.

Value

A ggplot object

References

Implementation of circular forest plot <https://mp.weixin.qq.com/s/PBKcsEFGGrDSQJp6ZmUgfHA>

See Also

Other br_show: [br_show_coxph_diagnostics\(\)](#), [br_show_fitted_line\(\)](#), [br_show_fitted_line_2d\(\)](#), [br_show_forest\(\)](#), [br_show_forest_ggstats\(\)](#), [br_show_forest_ggstatsplot\(\)](#), [br_show_nomogram\(\)](#), [br_show_residuals\(\)](#), [br_show_risk_network\(\)](#), [br_show_survival_curves\(\)](#), [br_show_table\(\)](#), [br_show_table_gt\(\)](#)

Examples

```

m <- br_pipeline(mtcars,
  y = "mpg",
  x = colnames(mtcars)[2:4],
  x2 = "vs",
  method = "gaussian"
)
br_show_forest_circle(m)
br_show_forest_circle(m, style = "bars")
br_show_forest_circle(m, sort_by = "estimate")
br_show_forest_circle(m, ref_line = FALSE)
br_show_forest_circle(m, ref_line = 0.5)

```

br_show_forest_comparison

Show forest plot for model comparison

Description**[Experimental]**

Creates a forest plot comparing univariate and multivariate model results side by side. Each variable shows estimates from both modeling approaches.

Usage

```
br_show_forest_comparison(comparison, ..., xlim = NULL, rm_controls = TRUE)
```

Arguments

comparison	A breg_comparison object from br_compare_models() .
...	Additional arguments passed to forestploter::forest() .
xlim	Numeric vector of length 2 specifying x-axis limits.
rm_controls	If TRUE, show only focal variables (default).

Value

A forest plot object.

See Also

Other br_compare: [br_compare_models\(\)](#)

Examples

```
lung <- survival::lung |>
  dplyr::filter(ph.ecog != 3)
lung$ph.ecog <- factor(lung$ph.ecog)

comparison <- br_compare_models(
  lung,
  y = c("time", "status"),
  x = c("ph.ecog", "ph.karno", "pat.karno"),
  x2 = c("age", "sex"),
  method = "coxph"
)

br_show_forest_comparison(comparison)
```

br_show_forest_ggstats

Show a forest plot with ggstats interface

Description**[Stable]**

Provides an interface to visualize the model results with **ggstats** package.

Usage

```
br_show_forest_ggstats(breg, idx = NULL, ...)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Index or names (focal variables) of the model(s).
...	Arguments passing to <code>ggstats::ggcoef_table()</code> or <code>ggstats::ggcoef_compare()</code> excepts model.

Value

A plot

See Also

Other `br_show`: `br_show_coxph_diagnostics()`, `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstatsplot()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`, `br_show_table_gt()`

Examples

```

if (rlang::is_installed("ggstats")) {
  m <- br_pipeline(mtcars,
    y = "mpg",
    x = colnames(mtcars)[2:4],
    x2 = "vs",
    method = "gaussian"
  )
  br_show_forest_ggstats(m)
}

```

br_show_forest_ggstatsplot

Show a forest plot with ggstatsplot interface

Description**[Stable]**

Provides an interface to visualize the model results with **ggstatsplot** package.

Usage

```
br_show_forest_ggstatsplot(breg, idx = 1, ...)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Length-1 vector. Index or name (focal variable) of the model. This is different from <code>idx</code> in <code>br_show_forest_ggstats</code> , only one model is supported to visualized here, so only length-1 vector is supported as <code>idx</code> .
...	Arguments passing to <code>ggstatsplot::ggcoefstats()</code> excepts <code>x</code> .

Value

A plot

See Also

Other `br_show`: `br_show_coxph_diagnostics()`, `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`, `br_show_table_gt()`

Examples

```
if (rlang::is_installed("ggstats")) {
  m <- br_pipeline(mtcars,
    y = "mpg",
    x = colnames(mtcars)[2:4],
    x2 = "vs",
    method = "gaussian"
  )
  br_show_forest_ggstatsplot(m)
}
```

br_show_nomogram

*Show nomogram for regression models***Description****[Experimental]**

Creates a nomogram (graphical calculator) for regression models, particularly useful for Cox proportional hazards models. A nomogram allows visual calculation of predicted outcomes by assigning points to variable values and summing them to get total points that correspond to predicted probabilities.

Usage

```
br_show_nomogram(
  breg,
  idx = NULL,
  time_points = c(12, 24, 36),
  fun_at = NULL,
  point_range = c(0, 100),
  title = NULL,
  subtitle = NULL
)
```

Arguments

breg	A breg object with fitted regression models.
idx	Index or name of the model to use for the nomogram. If NULL, uses the first model.
time_points	For Cox models, time points at which to show survival probabilities. Default is c(12, 24, 36) representing months.
fun_at	For non-survival models, the function values at which to show predictions.
point_range	Range of points to use in the nomogram scale. Default is c(0, 100).
title	Plot title. If NULL, generates automatic title.
subtitle	Plot subtitle.

Value

A ggplot2 object showing the nomogram.

See Also

Other br_show: `br_show_coxph_diagnostics()`, `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`, `br_show_table_gt()`

Examples

```
# Cox regression nomogram

lung <- survival::lung |> dplyr::filter(ph.ecog != 3)
lung$ph.ecog <- factor(lung$ph.ecog)
mds <- br_pipeline(
  lung,
  y = c("time", "status"),
  x = c("age", "ph.ecog"),
  x2 = "sex",
  method = "coxph"
)
p <- br_show_nomogram(mds)
p

# Linear regression nomogram
mds_lm <- br_pipeline(
  mtcars,
  y = "mpg",
  x = c("hp", "wt"),
  x2 = "vs",
  method = "gaussian"
)
p2 <- br_show_nomogram(mds_lm, fun_at = c(15, 20, 25, 30))
p2
```

br_show_residuals

*Show residuals vs fitted plot for regression models***Description****[Experimental]**

This function creates residual plots to diagnose model fit. It can display:

- Residuals vs fitted values plots for individual models
- Multiple residual plots when multiple models are selected
- Customizable plot appearance through ggplot2

Usage

```
br_show_residuals(breg, idx = NULL, plot_type = "fitted")
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Index or names (focal variables) of the model(s). If NULL (default), all models are included. If length-1, shows residuals for a single model. If length > 1, shows faceted plots for multiple models.
plot_type	Character string specifying the type of residual plot. Options: "fitted" (residuals vs fitted values, default), "qq" (Q-Q plot), "scale_location" (scale-location plot).

Value

A ggplot object

See Also

Other br_show: [br_show_coxph_diagnostics\(\)](#), [br_show_fitted_line\(\)](#), [br_show_fitted_line_2d\(\)](#), [br_show_forest\(\)](#), [br_show_forest_circle\(\)](#), [br_show_forest_ggstats\(\)](#), [br_show_forest_ggstatsplot\(\)](#), [br_show_nomogram\(\)](#), [br_show_risk_network\(\)](#), [br_show_survival_curves\(\)](#), [br_show_table\(\)](#), [br_show_table_gt\(\)](#)

Examples

```
m <- br_pipeline(mtcars,
  y = "mpg",
  x = colnames(mtcars)[2:4],
  x2 = "vs",
  method = "gaussian"
)

# Single model residual plot
br_show_residuals(m, idx = 1)

# Multiple models
br_show_residuals(m, idx = c(1, 2))

# All models
br_show_residuals(m)
```

br_show_risk_network *Show connected risk network plot*

Description**[Stable]****Usage**

br_show_risk_network(breg, ...)

Arguments

breg A regression object with results (must pass `assert_breg_obj_with_results()`).

... Arguments passing to [br_get_results\(\)](#) for subsetting data table.

Value

A plot

See Also

Other br_show: [br_show_coxph_diagnostics\(\)](#), [br_show_fitted_line\(\)](#), [br_show_fitted_line_2d\(\)](#), [br_show_forest\(\)](#), [br_show_forest_circle\(\)](#), [br_show_forest_ggstats\(\)](#), [br_show_forest_ggstatsplot\(\)](#), [br_show_nomogram\(\)](#), [br_show_residuals\(\)](#), [br_show_survival_curves\(\)](#), [br_show_table\(\)](#), [br_show_table_gt\(\)](#)

Other risk_network: [polar_connect\(\)](#), [polar_init\(\)](#)

Examples

```
lung <- survival::lung
# Cox-PH regression
mod_surv <- br_pipeline(
  data = lung,
  y = c("time", "status"),
  x = c("age", "ph.ecog", "ph.karno"),
  x2 = c("factor(sex)"),
  method = "coxph"
)

if (requireNamespace("ggnewscale")) {
  p <- br_show_risk_network(mod_surv)
  p
}
```

br_show_survival_curves

Show survival curves based on model scores

Description

[Experimental]

Generate survival curves by grouping observations based on model prediction scores. This function is specifically designed for Cox regression models and creates survival curves comparing different risk groups.

Usage

```
br_show_survival_curves(
  breg,
  idx = NULL,
  n_groups = 3,
  group_labels = NULL,
  title = NULL,
  subtitle = NULL
)
```

Arguments

breg	A breg object with fitted Cox regression models.
idx	Index or name of the model to use for prediction. If NULL, uses the first model.
n_groups	Number of groups to create based on score quantiles. Default is 3.
group_labels	Custom labels for the groups. If NULL, uses "Low", "Medium", "High" for 3 groups or "Q1", "Q2", etc. for other numbers.
title	Plot title. If NULL, generates automatic title.
subtitle	Plot subtitle.

Value

A ggplot2 object showing survival curves.

See Also

Other br_show: [br_show_coxph_diagnostics\(\)](#), [br_show_fitted_line\(\)](#), [br_show_fitted_line_2d\(\)](#), [br_show_forest\(\)](#), [br_show_forest_circle\(\)](#), [br_show_forest_ggstats\(\)](#), [br_show_forest_ggstatsplot\(\)](#), [br_show_nomogram\(\)](#), [br_show_residuals\(\)](#), [br_show_risk_network\(\)](#), [br_show_table\(\)](#), [br_show_table_gt\(\)](#)

Examples

```
# Cox regression example with survival curves
if (requireNamespace("survival", quietly = TRUE)) {
  lung <- survival::lung |> dplyr::filter(ph.ecog != 3)
  mds <- br_pipeline(
    lung,
    y = c("time", "status"),
    x = c("age", "ph.ecog"),
    x2 = "sex",
    method = "coxph"
  )
  p <- br_show_survival_curves(mds)
  print(p)
}
```

br_show_table	Show model tidy results in table format
---------------	---

Description**[Stable]****Usage**

```
br_show_table(
  breg,
  ...,
  args_table_format = list(),
  export = FALSE,
  args_table_export = list()
)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
...	Arguments passing to <code>br_get_results()</code> for subsetting table.
args_table_format	A list of arguments passing to <code>insight::format_table()</code> .
export	Logical. If TRUE, show table for export purpose, e.g., present the table in Mark-down or HTML format.
args_table_export	A list of arguments passing to <code>insight::export_table()</code> . Only works when export is TRUE.

Value

A table

See Also

Other br_show: `br_show_coxph_diagnostics()`, `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table_gt()`

Examples

```
m <- br_pipeline(mtcars,
  y = "mpg",
  x = colnames(mtcars)[2:4],
  x2 = "vs",
  method = "gaussian"
)

br_show_table(m)
br_show_table(m, export = TRUE)
if (interactive()) {
  br_show_table(m, export = TRUE, args_table_export = list(format = "html"))
}
```

br_show_table_gt

*Show regression models with gtsummary interface***Description****[Stable]**

Provides an interface to visualize the model results with **gtsummary** package in table format. check https://www.danielsjoberg.com/gtsummary/articles/tbl_regression.html#customize-output to see possible output customization.

Usage

```
br_show_table_gt(breg, idx = NULL, ..., tab_spanner = NULL)
```

Arguments

breg	A regression object with results (must pass <code>assert_breg_obj_with_results()</code>).
idx	Index or names (focal variables) of the model(s).
...	Arguments passing to <code>gtsummary::tbl_regression()</code> excepts x.
tab_spanner	(character) Character vector specifying the spanning headers. Must be the same length as tbls. The strings are interpreted with <code>gt::md</code> . Must be same length as tbls argument. Default is NULL, and places a default spanning header. If FALSE, no header will be placed.

Value

A table

See Also

Other `br_show`: `br_show_coxph_diagnostics()`, `br_show_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_circle()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_nomogram()`, `br_show_residuals()`, `br_show_risk_network()`, `br_show_survival_curves()`, `br_show_table()`

Examples

```
if (rlang::is_installed("gtsummary")) {  
  m <- br_pipeline(mtcars,  
    y = "mpg",  
    x = colnames(mtcars)[2:4],  
    x2 = "vs",  
    method = "gaussian"  
  )  
  br_show_table_gt(m)  
}
```

pipeline

Modeling and analysis pipeline

Description**[Stable]**

Provides a set of functions for running batch regression analysis. Combines data setup, model configuration, and execution steps into a single workflow. Supports both GLM and Cox-PH models with options for focal/control terms and parallel processing.

Usage

```
br_pipeline(  
  data,  
  y,  
  x,  
  method,  
  x2 = NULL,  
  group_by = NULL,  
  n_workers = 1L,  
  run_parallel = lifecycle::deprecated(),  
  dry_run = FALSE,  
  model_args = list(),  
  run_args = list(),
```

```

    filter_x = FALSE,
    filter_na_prop = 0.8,
    filter_sd_min = 1e-06,
    filter_var_min = 1e-06,
    filter_min_levels = 2
  )

br_set_y(obj, y)

br_set_x(
  obj,
  ...,
  filter_x = FALSE,
  filter_na_prop = 0.8,
  filter_sd_min = 1e-06,
  filter_var_min = 1e-06,
  filter_min_levels = 2
)

br_set_x2(obj, ...)

br_set_model(obj, method, ...)

br_run(
  obj,
  ...,
  group_by = NULL,
  n_workers = 1L,
  run_parallel = lifecycle::deprecated()
)

```

Arguments

<code>data</code>	A <code>data.frame</code> containing all necessary variables for analysis. Column names should follow R's naming conventions.
<code>y</code>	Character vector specifying dependent variables (response variables). For GLM models, this is typically a single character (e.g., "outcome"). For Cox-PH models, it should be a length-2 vector in the format <code>c("time", "status")</code> .
<code>x</code>	Character vector specifying focal independent terms (predictors).
<code>method</code>	Method for model construction. A name or a list specifying custom model setting. A string representing a complex method setting is acceptable, e.g., <code>'quasi(variance = "mu", link = "log")'</code> . Or a list with 4 elements, see br_avail_method_config() for examples.
<code>x2</code>	Character vector specifying control independent terms (predictors, optional).
<code>group_by</code>	A string specifying the group by column.
<code>n_workers, run_parallel</code>	Integer, indicating integer number of workers to launch, default is 1L. When >1, run modeling code in parallel in the background.

<code>dry_run</code>	If TRUE, generates modeling descriptions without executing the run. Use this to inspect the construction first.
<code>model_args</code>	A list of arguments passed to <code>br_set_model()</code> .
<code>run_args</code>	A list of arguments passed to <code>br_run()</code> .
<code>filter_x</code>	Logical, whether to enable pre-filtering of focal variables. Default is FALSE.
<code>filter_na_prop</code>	Numeric, maximum proportion of NA values allowed for a variable. Default is 0.8.
<code>filter_sd_min</code>	Numeric, minimum standard deviation required for a variable. Default is 1e-6.
<code>filter_var_min</code>	Numeric, minimum variance required for a variable. Default is 1e-6.
<code>filter_min_levels</code>	Numeric, minimum number of unique levels required for categorical variables. Default is 2.
<code>obj</code>	An object of class <code>breg</code> .
<code>...</code>	Additional arguments depending on the called function. • <code>br_set_x()</code> for passing focal terms as characters. • <code>br_set_x2()</code> for passing control terms as characters. • <code>br_set_model()</code> for passing other configurations for modeling. • <code>br_run()</code> for passing other configurations for obtaining modeling results with <code>broom.helpers::tidy_plus_plus()</code>. e.g., The default value for <code>exponentiate</code> is FALSE (coefficients are not exponentiated). For logistic, and Cox-PH regressions models, <code>exponentiate</code> is set to TRUE at default.

Details

Please note the difference between **variables** and **terms**, e.g., `x + poly(x, 2)` has *one* variable `x`, but *two* terms `x` and `poly(x, 2)`.

Global options:

breg supported global options can be set with `options()`. Currently they are used in `br_run()`.

- `breg.save_model`: If TRUE, saves models to local disk.
- `breg.path`: A path for saving models, defaults to using a temporary directory.

Value

An object of class `breg` with input values added to corresponding slot(s). For `br_run()`, the returned object is a `breg` object with results added to the slots `@results` and `@results_tidy`, note that `@models` is updated to a list of constructed model object (See [accessors](#)).

Functions

- `br_pipeline()`: All-in-one end to end wrapper to run the regression analysis in batch. Which could be splitted into the following steps
- `br_set_y()`: Set dependent variables for model construction.
- `br_set_x()`: Set focal terms for model construction.
- `br_set_x2()`: Set control terms for model construction (Optional in pipeline).
- `br_set_model()`: Set model configurations.
- `br_run()`: Run the regression analysis in batch.

See Also

[accessors](#) for accessing breg object properties.

Examples

```
library(bregr)
# 1. Pipeline -----
# 1.1. A single linear model -----
m <- breg(mtcars) |> # set model data
  br_set_y("mpg") |> # set dependent variable
  br_set_x("qsec") |> # set focal variables
  br_set_model("gaussian") |> # set model
  br_run() # run analysis

# get model tidy result
br_get_results(m, tidy = TRUE)
# or m@results_tidy

# compare with R's built-in function
lm(mpg ~ qsec, data = mtcars) |> summary()
# 1.2. Batch linear model -----
# control variables are injected in all constructed models
# focal variables are injected in constructed models one by one
m2 <- breg(mtcars) |>
  br_set_y("mpg") |>
  br_set_x(colnames(mtcars)[2:4]) |> # set focal variables
  br_set_x2("vs") |> # set control variables
  br_set_model("gaussian") |>
  br_run()

# 1.3. Group by model -----
m3 <- breg(mtcars) |>
  br_set_y("mpg") |>
  br_set_x("cyl") |>
  br_set_x2("wt") |> # set control variables
  br_set_model("gaussian") |>
  br_run(group_by = "am")

# 2. All-in-one pipeline wrapper ---
m4 <- br_pipeline(mtcars,
  y = "mpg",
  x = colnames(mtcars)[2:4],
  x2 = "vs",
  method = "gaussian"
)

# 3. Customized model -----
dt <- data.frame(x = rnorm(100))
dt$y <- rpois(100, exp(1 + dt$x))

m5 <- breg(dt) |>
```

```

br_set_y("y") |>
br_set_x("x") |>
br_set_model(method = 'quasi(variance = "mu", link = "log")') |>
br_run()

# 4. Non-standard variable names ---
# bregr automatically handles column names with special characters
# (., hyphens, leading digits, spaces, reserved words)
dt_special <- data.frame(
  y = rnorm(30),
  age = rnorm(30, 60, 10),
  g = rnorm(30)
)
colnames(dt_special)[3] <- "FGFR3::TACC3" # fusion gene notation
m6 <- br_pipeline(dt_special,
  y = "y", x = "FGFR3::TACC3", x2 = "age", method = "gaussian"
)
br_get_results(m6, tidy = TRUE)

```

polar_connect

Connects dots

Description

[Stable]

Check [polar_init\(\)](#) for examples.

Usage

```
polar_connect(data, mapping, ...)
```

Arguments

data	A <code>data.frame</code> contains connections of all events.
mapping	Set of aesthetic mappings to ggplot2::geom_segment() . Set mappings for x and xend are required.
...	Other arguments passing to ggplot2::geom_segment() .

Value

A `ggplot` object.

See Also

Other risk_network: [br_show_risk_network\(\)](#), [polar_init\(\)](#)

polar_init	<i>Init a dot plot in polar system</i>
------------	--

Description**[Stable]****Usage**

```
polar_init(data, mapping, ...)
```

Arguments

data	A data.frame contains all events, e.g., genes.
mapping	Set of aesthetic mappings to <code>ggplot2::geom_point()</code> . You should not set mapping for y.
...	Other arguments passing to <code>ggplot2::geom_point()</code> .

Value

A ggplot object.

See Also

Other risk_network: `br_show_risk_network()`, `polar_connect()`

Examples

```
library(ggplot2)
# -----
# Init a polar plot
# -----

data <- data.frame(x = LETTERS[1:7])

p1 <- polar_init(data, aes(x = x))
p1

# Set aes value
p2 <- polar_init(data, aes(x = x), size = 3, color = "red", alpha = 0.5)
p2

# Set aes mapping
set.seed(123L)
data1 <- data.frame(
  x = LETTERS[1:7],
  shape = c("r", "r", "r", "b", "b", "b", "b"),
  color = c("r", "r", "r", "b", "b", "b", "b"),
  size = abs(rnorm(7))
```

```

)
# Check https://ggplot2.tidyverse.org/reference/geom_point.html
# for how to use both stroke and color
p3 <- polar_init(data1, aes(x = x, size = size, color = color, shape = shape), alpha = 0.5)
p3

# -----
# Connect polar dots
# -----
data2 <- data.frame(
  x1 = LETTERS[1:7],
  x2 = c("B", "C", "D", "E", "C", "A", "C"),
  color = c("r", "r", "r", "b", "b", "b", "b")
)
p4 <- p3 + polar_connect(data2, aes(x = x1, xend = x2))
p4

p5 <- p3 + polar_connect(data2, aes(x = x1, xend = x2, color = color), alpha = 0.8, linetype = 2)
p5

# Use two different color scales
if (requireNamespace("ggnewscale")) {
  library(ggnewscale)
  p6 <- p3 +
    new_scale("color") +
    polar_connect(data2, aes(x = x1, xend = x2, color = color), alpha = 0.8, linetype = 2)
  p6 + scale_color_brewer()
  p6 + scale_color_manual(values = c("darkgreen", "magenta"))
}

```

print.breg

Print method for breg object

Description

[Stable]

Prints a summary of the breg object including slots, focal and control terms, group_by, and model configuration. Set raw = TRUE for the internal S7 representation.

print.breg_comparison *Print method for breg_comparison object*

Description

Print method for breg_comparison object

Usage

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

Arguments

x	A breg_comparison object.
...	Additional arguments (not used).

Index

- * **accessors**
 - accessors, [2](#)
 - br_diagnose, [8](#)
 - br_get_model_stats, [9](#)
 - br_predict, [10](#)
- * **br_compare**
 - br_compare_models, [6](#)
 - br_show_forest_comparison, [25](#)
- * **br_seed**
 - br_seed, [11](#)
 - br_seed_model, [13](#)
 - br_seed_screen, [15](#)
 - br_seed_select, [16](#)
- * **br_show**
 - br_show_coxph_diagnostics, [18](#)
 - br_show_fitted_line, [20](#)
 - br_show_fitted_line_2d, [21](#)
 - br_show_forest, [22](#)
 - br_show_forest_circle, [23](#)
 - br_show_forest_ggstats, [26](#)
 - br_show_forest_ggstatsplot, [27](#)
 - br_show_nomogram, [28](#)
 - br_show_residuals, [29](#)
 - br_show_risk_network, [31](#)
 - br_show_survival_curves, [32](#)
 - br_show_table, [33](#)
 - br_show_table_gt, [34](#)
- * **risk_network**
 - br_show_risk_network, [31](#)
 - polar_connect, [39](#)
 - polar_init, [40](#)

accessors, [2](#), [8–10](#), [37](#), [38](#)

avails, [4](#)

br_avail_method_config (avails), [4](#)

br_avail_method_config(), [36](#)

br_avail_methods (avails), [4](#)

br_avail_methods(), [5](#)

br_avail_methods_use_exp (avails), [4](#)

br_compare_models, [6](#), [25](#)

br_compare_models(), [25](#)

br_diagnose, [4](#), [8](#), [9](#), [10](#)

br_get_config (accessors), [2](#)

br_get_data (accessors), [2](#)

br_get_group_by (accessors), [2](#)

br_get_model (accessors), [2](#)

br_get_model_names (accessors), [2](#)

br_get_model_stats, [4](#), [8](#), [9](#), [10](#)

br_get_models (accessors), [2](#)

br_get_n_x (accessors), [2](#)

br_get_n_x2 (accessors), [2](#)

br_get_results (accessors), [2](#)

br_get_results(), [9](#), [31](#), [33](#)

br_get_x (accessors), [2](#)

br_get_x2 (accessors), [2](#)

br_get_y (accessors), [2](#)

br_pipeline (pipeline), [35](#)

br_pipeline(), [12](#), [15](#)

br_predict, [4](#), [8](#), [9](#), [10](#)

br_rename_models (accessors), [2](#)

br_run (pipeline), [35](#)

br_run(), [3](#), [7](#)

br_seed, [11](#), [14](#), [16](#), [17](#)

br_seed_model, [12](#), [13](#), [16](#), [17](#)

br_seed_screen, [12](#), [14](#), [15](#), [17](#)

br_seed_screen(), [17](#)

br_seed_select, [12](#), [14](#), [16](#), [16](#)

br_set_model (pipeline), [35](#)

br_set_model(), [5](#), [7](#)

br_set_x (pipeline), [35](#)

br_set_x2 (pipeline), [35](#)

br_set_y (pipeline), [35](#)

br_show_coxph_diagnostics, [18](#), [20](#), [21](#), [23](#), [24](#), [26](#), [27](#), [29–32](#), [34](#), [35](#)

br_show_fitted_line, [19](#), [20](#), [21](#), [23](#), [24](#), [26](#), [27](#), [29–32](#), [34](#), [35](#)

br_show_fitted_line(), [21](#)

br_show_fitted_line_2d, [19](#), [20](#), [21](#), [23](#), [24](#),

26, 27, 29–32, 34, 35
 br_show_forest, 19–21, 22, 24, 26, 27,
 29–32, 34, 35
 br_show_forest(), 23
 br_show_forest_circle, 19–21, 23, 23, 26,
 27, 29–32, 34, 35
 br_show_forest_comparison, 7, 25
 br_show_forest_ggstats, 19–21, 23, 24, 26,
 27, 29–32, 34, 35
 br_show_forest_ggstatsplot, 19–21, 23,
 24, 26, 27, 29–32, 34, 35
 br_show_nomogram, 19–21, 23, 24, 26, 27, 28,
 30–32, 34, 35
 br_show_residuals, 19–21, 23, 24, 26, 27,
 29, 29, 31, 32, 34, 35
 br_show_risk_network, 19–21, 23, 24, 26,
 27, 29, 30, 31, 32, 34, 35, 39, 40
 br_show_survival_curves, 19–21, 23, 24,
 26, 27, 29–31, 32, 34, 35
 br_show_table, 19–21, 23, 24, 26, 27, 29–32,
 33, 35
 br_show_table_gt, 19–21, 23, 24, 26, 27,
 29–32, 34, 34
 breg, 5
 broom.helpers::tidy_plus_plus(), 3, 6,
 37
 broom::tidy(), 3, 6

 dplyr::filter(), 3

 forestploter::forest(), 22, 23, 25

 ggplot2::coord_polar(), 23
 ggplot2::geom_point(), 40
 ggplot2::geom_segment(), 39
 ggstats::ggcoef_compare(), 26
 ggstats::ggcoef_table(), 26
 ggstatsplot::ggcoefstats(), 27
 gtsummary::tbl_regression(), 34

 insight::export_table(), 33
 insight::format_table(), 33

 pipeline, 4, 5, 35
 polar_connect, 31, 39, 40
 polar_init, 31, 39, 40
 polar_init(), 39
 print.breg, 41
 print.breg_comparison, 41

 survival::cox.zph, 19

 visreg::visreg(), 20
 visreg::visreg2d(), 21