

# Package ‘bregr’

July 22, 2025

**Title** Easy and Efficient Batch Processing of Regression Models

**Version** 1.0.0

**Maintainer** Shixiang Wang <w\_shixiang@163.com>

**Description** Easily process batches of univariate or multivariate regression models. Returns results in a tidy format and generates visualization plots for straightforward interpretation (Wang, Shixiang, et al. (2021) <[DOI:10.48550/arXiv.2110.14232](https://doi.org/10.48550/arXiv.2110.14232)>).

**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, parallel, purrr, rlang (>= 1.1.0), S7, survival, tibble, utils, vctrs (>= 0.5.0)

**Suggests** ggnewscale, ggstats, ggstatsplot, gtsummary, knitr, rmarkdown, testthat (>= 3.0.0), visreg

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**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** 2025-06-28 17:00:02 UTC

## Contents

accessors	2
avails	4
breg	5
br_show_fitted_line	6
br_show_fitted_line_2d	7
br_show_forest	8
br_show_forest_ggstats	9
br_show_forest_ggstatsplot	10
br_show_risk_network	11
br_show_table	12
br_show_table_gt	13
pipeline	14
polar_connect	16
polar_init	17
print.breg	19
<b>Index</b>	<b>20</b>

---

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)`

```
br_get_model(obj, idx)
```

```
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.
<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 constructed models.
- `br_get_model()` returns a subset of constructed models.
- `br_get_results()` returns modeling result `data.frame`.

### See Also

[pipeline](#) for building breg objects.

### 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_model(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

### [Experimental]

Package resource, definitions ready for use.

## Usage

```
br_avail_methods()
br_avail_methods_use_exp()
```

## 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 **bregr**.

## 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 data.frame 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 data.frame of model results from <code>broom.helpers::tidy_plus_plus()</code> .
results_tidy	A data.frame of tidy model results from <code>broom::tidy()</code> .

## Value

A constructed breg object.

## Examples

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

---

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 fit and data.

## Value

A plot

## See Also

Other `br_show`: `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsplot()`, `br_show_risk_network()`, `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

<code>breg</code>	A regression object with results (must pass <code>assert_breg_obj_with_results()</code> ).
<code>idx</code>	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> .
<code>...</code>	Arguments passing to <code>visreg::visreg2d()</code> excepts fit and data.

## Value

A plot

## See Also

Other `br_show`: `br_show_fitted_line()`, `br_show_forest()`, `br_show_forest_ggstats()`, `br_show_forest_ggstatsp`, `br_show_risk_network()`, `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	<i>Show a forest plot for regression results</i>
----------------	--

---

## 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
)
```

## 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 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.
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).

## Value

A plot



## See Also

Other `br_show`: [br\\_show\\_fitted\\_line\(\)](#), [br\\_show\\_fitted\\_line\\_2d\(\)](#), [br\\_show\\_forest\\_ggstats\(\)](#), [br\\_show\\_forest\\_ggstatsplot\(\)](#), [br\\_show\\_risk\\_network\(\)](#), [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\_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 <code>model</code> .

## Value

A plot

## See Also

Other `br_show`: [br\\_show\\_fitted\\_line\(\)](#), [br\\_show\\_fitted\\_line\\_2d\(\)](#), [br\\_show\\_forest\(\)](#), [br\\_show\\_forest\\_ggstatsplot\(\)](#), [br\\_show\\_risk\\_network\(\)](#), [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_fitted_line()`, `br_show_fitted_line_2d()`, `br_show_forest()`, `br_show_forest_ggstats`, `br_show_risk_network()`, `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\_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\\_fitted\\_line\(\)](#), [br\\_show\\_fitted\\_line\\_2d\(\)](#), [br\\_show\\_forest\(\)](#), [br\\_show\\_forest\\_ggstatsplot\(\)](#), [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"
```

```

)
p <- br_show_risk_network(mod_surv)
p

```

---

br_show_table	<i>Show model tidy results in table format</i>
---------------	--

---

## 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 Markdown 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\\_fitted\\_line\(\)](#), [br\\_show\\_fitted\\_line\\_2d\(\)](#), [br\\_show\\_forest\(\)](#), [br\\_show\\_forest\\_ggstats](#), [br\\_show\\_forest\\_ggstatsplot\(\)](#), [br\\_show\\_risk\\_network\(\)](#), [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****[Experimental]**

Provides an interface to visualize the model results with **gtsummary** package in table format. check [https://www.danieldsjoberg.com/gtsummary/articles/tbl\\_regression.html#customize-output](https://www.danieldsjoberg.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 <code>tbls</code> . The strings are interpreted with <code>gt::md</code> . Must be same length as <code>tbls</code> argument. Default is <code>NULL</code> , and places a default spanning header. If <code>FALSE</code> , no header will be placed.

**Value**

A table

**See Also**

Other `br_show`: [br\\_show\\_fitted\\_line\(\)](#), [br\\_show\\_fitted\\_line\\_2d\(\)](#), [br\\_show\\_forest\(\)](#), [br\\_show\\_forest\\_ggstats](#), [br\\_show\\_forest\\_ggstatsplot\(\)](#), [br\\_show\\_risk\\_network\(\)](#), [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,  
  run_parallel = 1L,  
  model_args = list(),  
  run_args = list()  
)  
  
br_set_y(obj, y)  
  
br_set_x(obj, ...)  
  
br_set_x2(obj, ...)  
  
br_set_model(obj, method, ...)  
  
br_run(obj, ..., group_by = NULL, run_parallel = 1L)
```

## 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 string representing a complex method setting is acceptable, e.g., <code>'quasi(variance = "mu", link = "log")'</code> .
<code>x2</code>	Character vector specifying control independent terms (predictor, optional).
<code>group_by</code>	A string specifying the group by column.
<code>run_parallel</code>	Integer, indicating cores to run the task, default is 1.
<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>obj</code>	An object of class <code>breg</code> .
<code>...</code>	Additional arguments depending on the called function. <ul style="list-style-type: none"> <li>• <code>br_set_x()</code> for passing focal terms as characters.</li> <li>• <code>br_set_x2()</code> for passing control terms as characters.</li> <li>• <code>br_set_model()</code> for passing other configurations for modeling.</li> <li>• <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 <code>FALSE</code> (coefficients are not exponentiated). For logistic, and Cox-PH regressions models, <code>exponentiate</code> is set to <code>TRUE</code> at default.</li> </ul>

## 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)`.

## 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"
)

```



**Description****[Stable]**Check [polar\\_init\(\)](#) for examples.**Usage**`polar_connect(data, x1, x2, ...)`**Arguments**

`data` A data.frame contains connections of all events.

`x1, x2` Column names (**without quote in aes()**) storing connected events.

`...` Arguments passing to `ggplot2::geom_segment()`, expect `c(x, xend, y, yend)` these 4 mapping parameters.

**Value**

A ggplot object.

**See Also**Other risk\_network: [br\\_show\\_risk\\_network\(\)](#), [polar\\_init\(\)](#)

---

`polar_init`*Init a dot plot in polar system*

---

**Description****[Stable]****Usage**`polar_init(data, x, ...)`**Arguments**

`data` A data.frame contains all events, e.g., genes.

`x` Column name (without quote) storing event list.

`...` Arguments passing to `ggplot2::geom_point()`.

**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, x = x)
p1

# Set aes value
p2 <- polar_init(data, 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, x = x, aes(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, x1, x2)
p4

# Unlike polar_init, mappings don't need to be included in aes()
p5 <- p3 + polar_connect(data2, x1, 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, x1, x2, color = color, alpha = 0.8, linetype = 2)
  p6 + scale_color_brewer()
  p6 + scale_color_manual(values = c("darkgreen", "magenta"))
}

```

---

<code>print.breg</code>	<i>Print method for breg object</i>
-------------------------	-------------------------------------

---

### **Description**

**[Experimental]**

### **Arguments**

<code>x</code>	An object of class <code>breg</code> .
<code>...</code>	Additional arguments (currently not used).
<code>raw</code>	Logical, whether to print raw S7 representation. Default is <code>FALSE</code> .

### **Value**

Invisibly returns `x`.

# Index

- \* **br\_show**
  - br\_show\_fitted\_line, 6
  - br\_show\_fitted\_line\_2d, 7
  - br\_show\_forest, 8
  - br\_show\_forest\_ggstats, 9
  - br\_show\_forest\_ggstatsplot, 10
  - br\_show\_risk\_network, 11
  - br\_show\_table, 12
  - br\_show\_table\_gt, 13
- \* **risk\_network**
  - br\_show\_risk\_network, 11
  - polar\_connect, 16
  - polar\_init, 17

accessors, 2, 15, 16

avails, 4

br\_avail\_methods (avails), 4

br\_avail\_methods\_use\_exp (avails), 4

br\_get\_config (accessors), 2

br\_get\_data (accessors), 2

br\_get\_group\_by (accessors), 2

br\_get\_model (accessors), 2

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(), 11, 12

br\_get\_x (accessors), 2

br\_get\_x2 (accessors), 2

br\_get\_y (accessors), 2

br\_pipeline (pipeline), 14

br\_run (pipeline), 14

br\_run(), 3

br\_set\_model (pipeline), 14

br\_set\_model(), 4

br\_set\_x (pipeline), 14

br\_set\_x2 (pipeline), 14

br\_set\_y (pipeline), 14

br\_show\_fitted\_line, 6, 7, 9–13

br\_show\_fitted\_line(), 7

br\_show\_fitted\_line\_2d, 6, 7, 9–13

br\_show\_forest, 6, 7, 8, 9–13

br\_show\_forest\_ggstats, 6, 7, 9, 9, 10–13

br\_show\_forest\_ggstatsplot, 6, 7, 9, 10, 11–13

br\_show\_risk\_network, 6, 7, 9, 10, 11, 12, 13, 17

br\_show\_table, 6, 7, 9–11, 12, 13

br\_show\_table\_gt, 6, 7, 9–12, 13

breg, 5

broom.helpers::tidy\_plus\_plus(), 3, 5, 15

broom::tidy(), 3, 5

dplyr::filter(), 3

ggplot2::geom\_point(), 17

ggplot2::geom\_segment(), 17

ggstats::ggcoef\_compare(), 9

ggstats::ggcoef\_table(), 9

ggstatsplot::ggcoefstats(), 10

gtsummary::tbl\_regression(), 13

insight::export\_table(), 12

insight::format\_table(), 12

pipeline, 3, 4, 14

polar\_connect, 11, 16, 17

polar\_init, 11, 17, 17

polar\_init(), 17

print.breg, 19

visreg::visreg(), 6

visreg::visreg2d(), 7