# Package 'bregr'

June 28, 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>).

**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` *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

| | |
|---|---|
| obj | A breg object. |
| idx | Index or names (focal variables) of the model(s) to return. |
| tidy | If TRUE return tidy (compact) results, otherwise return comprehensive results. The tidy results are obtained from broom::tidy() while comprehensive results are obtained from broom.helpers::tidy_plus_plus(). The results can be configured when run with br_run(). |
| ... | Subset operations passing to dplyr::filter() 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` *Creates a new breg-class object*

## 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 `broom.helpers::tidy_plus_plus()`. |
| `results_tidy` | A `data.frame` of tidy model results from `broom::tidy()`. |

## 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 assert_breg_obj_with_results()). |
| idx | Length-1 vector. Index or name (focal variable) of the model. This is different from idx in br_show_forest_ggstats, only one model is supported to visualized here, so only length-1 vector is supported as idx. |
| ... | Arguments passing to visreg::visreg() 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

| | |
|---|---|
| `breg` | A regression object with results (must pass `assert_breg_obj_with_results()`). |
| `idx` | Length-1 vector. Index or name (focal variable) of the model. This is different from `idx` in br_show_forest_ggstats, only one model is supported to visualized here, so only length-1 vector is supported as `idx`. |
| `...` | Arguments passing to `visreg::visreg2d()` 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 *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
)
```

## Arguments

| | |
|---|---|
| `breg` | A regression object with results (must pass `assert_breg_obj_with_results()`). |
| `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 `forestploter::forest()`, run `vignette("forestploter-post", "forestploter")` to see more plot options. |
| `subset` | Expression for subsetting the results data (`br_get_results(breg)`). |
| `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 assert_breg_obj_with_results()). |
| idx | Index or names (focal variables) of the model(s). |
| ... | Arguments passing to ggstats::ggcoef_table() or ggstats::ggcoef_compare() excepts model. |

## Value

A plot

## See Also

Other br_show: br_show_fitted_line(), br_show_fitted_line_2d(), br_show_forest(), br_show_forest_ggstatsp
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 assert_breg_obj_with_results()). |
| idx | Length-1 vector. Index or name (focal variable) of the model. This is different from idx in br_show_forest_ggstats, only one model is supported to visualized here, so only length-1 vector is supported as idx. |
| ... | Arguments passing to ggstatsplot::ggcoefstats() excepts x. |

## 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_ggstats(`
`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 *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 `assert_breg_obj_with_results()`). |
| ... | Arguments passing to `br_get_results()` for subsetting table. |
| args_table_format | A list of arguments passing to `insight::format_table()`. |
| 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 `insight::export_table()`. 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 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 assert_breg_obj_with_results()). |
| idx | Index or names (focal variables) of the model(s). |
| ... | Arguments passing to gtsummary::tbl_regression() excepts x. |
| tab_spanner | (character)<br>Character vector specifying the spanning headers. Must be the same length as tbls. The strings are interpreted with gt::md. 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_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

| | |
|---|---|
| data | A data.frame containing all necessary variables for analysis. Column names should follow R's naming conventions. |
| y | 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 c("time", "status"). |
| x | Character vector specifying focal independent terms (predictors). |
| method | Method for model construction. A string representing a complex method setting is acceptable, e.g., 'quasi(variance = "mu", link = "log")'. |
| x2 | Character vector specifying control independent terms (predictor, optional). |
| group_by | A string specifying the group by column. |
| run_parallel | Integer, indicating cores to run the task, default is 1. |
| model_args | A list of arguments passed to br_set_model(). |
| run_args | A list of arguments passed to br_run(). |
| obj | An object of class breg. |
| ... | Additional arguments depending on the called function. |

- br_set_x() for passing focal terms as characters.
- br_set_x2() for passing control terms as characters.
- br_set_model() for passing other configurations for modeling.
- br_run() for passing other configurations for obtaining modeling results with broom.helpers::tidy_plus_plus(). e.g., The default value for exponentiate is FALSE (coefficients are not exponentiated). For logistic, and Cox-PH regressions models, exponentiate 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).

## 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"
)
```

---

`polar_connect` *Connects dots*

---

## 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"))
}
```

print.breg *Print method for breg object*

## Description

**[Experimental]**

## Arguments

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

## Value

Invisibly returns x.

# Index