Package 'bregr'

August 19, 2025

Title Easy and Efficient Batch Processing of Regression Models

Version 1.2.0

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. (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, 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, qs, rmarkdown, testthat (>= 3.0.0), UCSCXenaShiny, 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 2025-08-19 11:10:13 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, 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

obj	A breg object.
idx	Index or names (focal variables) of the model(s) to return. Default returns all.
auto_drop	If TRUE, automatically drop the list if only one model is selected.
new_names	Character vector to replace existing model names.
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 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_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 **bregr**.
- `br_avail_method_config()`: Returns model configs for specified method to generate modeling templates.

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_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 assert_breg_obj_with_results()).
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_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_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()`

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_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 **bregr**.

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 `assert_breg_obj_with_results()`).
`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 survival::cox.zph.

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 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_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 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_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 `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 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 `forestploter::forest()`, run `vignette("forestploter-post", "forestploter")` to see more plot options. For example, use `ticks_at` to specify custom ticks, generally a vector of 4-5 elements.
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).
log_first	Log transformed the estimates and their confident intervals. For only log scaled axis of the forest, use `x_trans = "log"`.

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/PBKcsEFGrDSQJp6ZmUgfHA

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_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_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 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_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 assert_breg_obj_with_results()).
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"
)
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 `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_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.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`) 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_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

`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 name or a list specifying custom model setting. A string representing a complex method setting is acceptable, e.g., 'quasi(variance = "mu", link = "log")'. Or a list with 4 elements, see `br_avail_method_config()` for examples.
`x2`	Character vector specifying control independent terms (predictors, optional).
`group_by`	A string specifying the group by column.
`n_workers, run_parallel`	Integer, indicating integer number of workers to launch, default is `1L`. When `>1`, run modeling code in parallel in the background.

dry_run	If TRUE, generates modeling descriptions without executing the run. Use this to inspect the construction first.
model_args	A list of arguments passed to br_set_model().
run_args	A list of arguments passed to br_run().
filter_x	Logical, whether to enable pre-filtering of focal variables. Default is FALSE.
filter_na_prop	Numeric, maximum proportion of NA values allowed for a variable. Default is 0.8.
filter_sd_min	Numeric, minimum standard deviation required for a variable. Default is 1e-6.
filter_var_min	Numeric, minimum variance required for a variable. Default is 1e-6.
filter_min_levels	Numeric, minimum number of unique levels required for categorical variables. Default is 2.
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).

Global options:

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

- bregr.save_model: If TRUE, saves models to local disk.
- bregr.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 -----------
## Not run:
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()

## End(Not run)
```

`polar_connect` *Connects dots*

Description

[Stable]

Check `polar_init()` for examples.

Usage

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

Arguments

`data`	A `data.frame` 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` *Init a dot plot in polar system*

Description

[Stable]

Usage

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

Arguments

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

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