Package 'curves'

May 11, 2026

Type Package

Title Model-Agnostic Response Curves for Fitted Models

Version 0.4.0

Description Create model-agnostic response-curve diagnostics for fitted prediction models. Supports profile curves, partial dependence, individual conditional expectation, and accumulated local effects; univariate curves, bivariate surfaces, ensemble summaries across multiple models, ALE-based interaction ranking, and optional raster-linked exploration with 'terra' and 'shiny'. Static displays are returned as 'ggplot2' plots. For more details on the methods see Molnar (2025) <https://christophm.github.io/interpretable-ml-book/>.

URL https://github.com/rvalavi/curves

BugReports https://github.com/rvalavi/curves/issues

Maintainer Roozbeh Valavi <valavi.r@gmail.com>

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0)

Imports cowplot, ggplot2 (>= 3.3.6)

Suggests disdat, knitr, mgcv, plotly, randomForest, rmarkdown, shiny, terra (>= 1.6-41), testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Roozbeh Valavi [aut, cre] (ORCID: <https://orcid.org/0000-0003-2495-5277>)

Repository CRAN

Date/Publication 2026-05-11 19:00:02 UTC

Contents

bivariate 2
interactions 5
mapcurve 7
multimodel 11
univariate 14

Index **18**

`bivariate` *Bivariate model-agnostic response surfaces*

Description

Plot how a fitted model's predictions change across pairs of predictors using reference-profile, partial dependence, or second-order accumulated local effects surfaces.

Usage

```
bivariate(
  model,
  x = NULL,
  predict_data = NULL,
  pairs = NULL,
  top_n = NULL,
  fun = NULL,
  ...,
  n = 40,
  background_n = n,
  extrapolate = FALSE,
  rug = FALSE,
  plot_type = c("heatmap", "contour", "surface"),
  zlab = "Prediction",
  bins = 8,
  palette = "viridis",
  response = NULL,
  nrows = NULL,
  ncols = NULL,
  method = c("pdp", "ale", "profile")
)
```

Arguments

`model`	A fitted model object that supports prediction.
`x`	A data frame or raster containing predictor variables. If `predict_data` is provided, this argument is ignored.

predict_data	A data frame containing values at which predictions should be made. If NULL, x must be provided.
pairs	Optional specification of predictor pairs to plot. Supply NULL to plot all unique pairs, a character or numeric vector of length 2 for a single pair, or a list/data frame/matrix of pairs. Numeric pairs are interpreted as predictor column indices.
top_n	Optional integer limit on the number of ALE pairs to plot after ranking them from highest to lowest interaction strength with interactions(). This is only supported when method = "ale" and can be used with pairs = NULL to keep, for example, only the top 5 ranked interaction surfaces.
fun	A function used to generate predictions from the model. If NULL, the generic predict() is used.
...	Additional arguments passed to fun.
n	Integer, number of points to sample for each numeric predictor variable (default: 40). For "ale", n sets the maximum number of intervals used to estimate local effects for each numeric predictor and defaults to 10, since the second-order surface uses an n x n cell grid and finer grids tend to leave most cells unsupported by data.
background_n	Integer, number of randomly sampled background rows used for "pdp" (default: n).
extrapolate	Logical, whether method = "ale" should display interpolated values for unsupported grid cells that contain no observations (default: FALSE). Ignored for other methods.
rug	Logical, whether to add a marginal rug for numeric predictor pairs in static plots (default: FALSE, but TRUE for method = "ale" so users can see which cells are supported by data).
plot_type	Character, plot type. Use "heatmap" for a static surface, "contour" for filled contours, or "surface" for an interactive 3D surface. The 3D surface requires the suggested plotly package and a single numeric predictor pair.
zlab	Character, label for the response legend or z-axis (default: "Prediction").
bins	Integer, number of contour bins for "contour" plots.
palette	Either a viridis option name (default: "viridis") or a character vector of colours used for the response scale.
response	Optional column name or index to select when fun returns multiple predictions per row. If NULL and exactly two prediction columns are returned, the second column is used.
nrows	Integer, number of rows in the plot grid. If NULL, it is automatically determined.
ncols	Integer, number of columns in the plot grid. If NULL, it is automatically determined.
method	Character, the surface type to plot. "profile" uses a single reference profile, "pdp" averages over sampled predictor rows, and "ale" draws a centred second-order accumulated local effects surface for numeric predictor pairs. Non-numeric pairs are ignored with a warning for "ale".

Details

Let $\hat{f}$ denote the fitted prediction function, let $S = \{a, b\}$ be the two plotted predictors, and let x_C contain the remaining predictors. The `"profile"` method evaluates a single reference profile over the two-dimensional grid,

$$g_S(z_a, z_b) = \hat{f}(z_a, z_b, x_C^{ref}).$$

This is a direct response surface around the mean/modal reference row.

For `"pdp"`, the surface is the Monte Carlo partial dependence estimate over m sampled background rows,

$$\hat{f}_{S,PDP}(z_a, z_b) = \frac{1}{m} \sum_{i=1}^{m} \hat{f}(z_a, z_b, x_C^{(i)}).$$

This marginalizes over the non-plotted predictors as described for PDPs by Molnar (2025). The result includes interactions with other predictors, but it may evaluate uncommon predictor combinations when predictors are dependent.

For `"ale"`, both predictors must be numeric. The two features are divided into rectangular cells with x-breaks $z_0 < \cdots < z_K$ and y-breaks $w_0 < \cdots < w_L$. For observations in cell (k, l), the second-order local effect is the mean corner contrast

$$\Delta_{k,l} = \frac{1}{n_{k,l}} \sum_{i \in N(k,l)} [\hat{f}(z_k, w_l, x_C^{(i)}) - \hat{f}(z_{k-1}, w_l, x_C^{(i)}) - \hat{f}(z_k, w_{l-1}, x_C^{(i)}) + \hat{f}(z_{k-1}, w_{l-1}, x_C^{(i)})].$$

The cell effects are accumulated over the grid using a half-cell correction that places each value at the cell centre, then centred by removing the (count-weighted) row, column, and overall means. The resulting surface is a centred second-order ALE estimate: it shows the additional **interaction effect** of the two predictors after their first-order main effects have been removed. Positive values indicate predictor combinations where the model response is higher than would be expected from adding the two first-order ALE effects alone, negative values indicate combinations where the joint effect is lower than that additive expectation, and values near zero indicate little additional interaction. For GLMs and other models with a nonlinear inverse link, this interpretation depends on the prediction scale: a model that is additive on the link scale can still show non-zero second-order ALE on the response scale. For example, a binomial `glm` without explicit interaction terms can still produce bivariate ALE structure when `type = "response"` because the inverse-logit is nonlinear; use `type = "link"` to inspect interaction in the linear predictor instead.

Cells that contain no observations are treated as unsupported: the cell effect is interpolated from neighbouring cells so the accumulation can run, but, by default, the cell is masked (`NA`) in the returned surface and rendered in `na.value` (default light grey) in the plot. This avoids extrapolating the interaction surface into regions of feature space that are not represented by the data, which is critical when predictors are correlated. Set `extrapolate = TRUE` to plot the interpolated values for unsupported cells instead.

Value

A `ggplot2` object for static plot types or a `plotly` widget for `plot_type = "surface"`.

References

Molnar, C. (2025). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (3rd ed.). https://christophm.github.io/interpretable-ml-book/

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189-1232.

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B*, 82(4), 1059-1086.

Examples

```
data(iris)
model <- lm(
  Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
  data = iris
)
response_plot <- bivariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")]
)
print(response_plot)

pdp_plot <- bivariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")],
  pairs = c("Sepal.Width", "Petal.Length"),
  method = "pdp",
  n = 25,
  background_n = 50,
  rug = TRUE
)
print(pdp_plot)

ale_plot <- bivariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")],
  pairs = c("Sepal.Width", "Petal.Length"),
  method = "ale",
  n = 10
)
print(ale_plot)

if (requireNamespace("plotly", quietly = TRUE)) {
  surface_plot <- bivariate(
    model,
    x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")],
    pairs = c("Sepal.Width", "Petal.Length"),
    plot_type = "surface"
  )
  surface_plot
}
```

`interactions` *Rank pairwise interactions using second-order ALE*

Description

Quantify and rank pairwise interaction strength using the same centred second-order accumulated local effects (ALE) surfaces used by `bivariate()` with `method = "ale"`. This is also the ranking used internally by `bivariate(method = "ale", top_n = ...)` when it selects the strongest surfaces to plot.

Usage

```
interactions(
  model,
  x = NULL,
  predict_data = NULL,
  pairs = NULL,
  fun = NULL,
  ...,
  n = 10,
  response = NULL,
  details = FALSE
)
```

Arguments

`model`	A fitted model object that supports prediction.
`x`	A data frame or raster containing predictor variables. If `predict_data` is provided, this argument is ignored.
`predict_data`	A data frame containing values at which predictions should be made. If `NULL`, `x` must be provided.
`pairs`	Optional specification of predictor pairs to rank. Supply `NULL` to rank all unique pairs, a character or numeric vector of length 2 for a single pair, or a list/data frame/matrix of pairs. Numeric pairs are interpreted as predictor column indices. Non-numeric pairs are ignored with a warning because second-order ALE currently supports numeric predictor pairs only.
`fun`	A function used to generate predictions from the model. If `NULL`, the generic `predict()` is used.
`...`	Additional arguments passed to `fun`.
`n`	Integer, maximum number of ALE intervals per numeric predictor (default: `10`).
`response`	Optional column name or index to select when `fun` returns multiple predictions per row. If `NULL` and exactly two prediction columns are returned, the second column is used.
`details`	Logical; if `FALSE` (default), return the ranking data frame. If `TRUE`, return a list containing the ranking plus the ALE tables and pair specifications used internally by `bivariate()` when `top_n` filters ALE surfaces. The returned tables keep interpolated values for unsupported cells so callers such as `bivariate()` can decide whether to mask or show them.

Details

Let $\hat{f}_{ab,ALE}(c_k, d_l)$ denote the centred second-order ALE surface for predictor pair (a, b) evaluated at cell centre (c_k, d_l), and let $n_{k,l}$ be the number of observed rows in that ALE cell. The returned interaction strength is the count-weighted root-mean-square ALE magnitude,

$$I_{ab} = \sqrt{\frac{\sum_{k,l} n_{k,l} \hat{f}_{ab,ALE}(c_k, d_l)^2}{\sum_{k,l} n_{k,l}}}.$$

This uses the observed ALE cell counts as weights, so densely supported parts of the predictor space contribute more than sparse cells. Additive predictor pairs score zero, and larger values indicate stronger non-additive joint effects. Cells with zero observations are excluded from the score.

Value

If `details = FALSE`, a data frame ordered from highest to lowest interaction strength, with columns `rank`, `pair`, `predictor_1`, `predictor_2`, `strength`, `support`, `supported_cells`, and `total_cells`. If `details = TRUE`, a list with components `ranking`, `pair_specs`, and `tables`.

References

Molnar, C. (2025). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (3rd ed.). https://christophm.github.io/interpretable-ml-book/

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B*, 82(4), 1059-1086.

Examples

```
data(iris)
model <- lm(
  Sepal.Length ~ Sepal.Width * Petal.Length + Petal.Width,
  data = iris
)

interactions(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")]
)
```

`mapcurve` *Interactive map-linked response curve explorer*

Description

Launch a Shiny app that links a prediction raster to model-agnostic univariate response curves. Clicking a map cell extracts that cell's predictor values and marks them on the curve panels, making spatial predictions and curve-based model behaviour inspectable together.

Usage

```
mapcurve(
  model,
  map,
  predictors,
  predict_data = NULL,
  fun = NULL,
  ...,
  n = 100,
  background_n = n,
  interval = c("none", "quantile"),
  interval_level = 0.8,
  ylab = "Prediction",
  rug = TRUE,
  ylim = NULL,
  color = "#ff5a36",
  response = NULL,
  nrows = NULL,
  ncols = 2,
  method = c("pdp", "ice", "ice+pdp", "ale", "profile"),
  selected_color = "deepskyblue3",
  show_selected_ice = TRUE,
  crosshair = TRUE,
  map_palette = grDevices::hcl.colors(64, "Inferno"),
  map_title = "Prediction map",
  launch = interactive()
)
```

Arguments

`model`	A fitted model object that supports prediction.
`map`	A single-layer `terra::SpatRaster` containing the predicted surface shown on the map.
`predictors`	A `terra::SpatRaster` containing the predictor layers used to extract covariate values at the clicked map cell.
`predict_data`	Optional data frame or raster used to build the response curves. If `NULL`, `predictors` is used.
`fun`	A function used to generate predictions from the model. If `NULL`, the generic `predict()` is used.
`...`	Additional arguments passed to `fun`.
`n`	Integer, number of points to sample for each numeric predictor variable (default: 100). For `"ale"`, `n` sets the maximum number of intervals used to estimate local effects for numeric predictors.
`background_n`	Integer, number of randomly sampled background rows used for `"pdp"`, `"ice"`, and `"ice+pdp"` (default: `n`).

interval	Character, interval type used to draw a PDP ribbon for numeric predictors. Only "quantile" is currently supported and only when method = "pdp". Defaults to "none".
interval_level	Numeric in (0, 1) giving the central quantile width used when interval = "quantile". Ignored otherwise.
ylab	Character, label for the response scale (default: "Prediction").
rug	Logical, whether to include rug marks for numeric predictors (default: TRUE).
ylim	Numeric vector of length 2, specifying the limits of the response axis. If NULL, limits are computed from the fitted curves.
color	Character, colour of the fitted response curves.
response	Optional column name or index to select when fun returns multiple predictions per row. If NULL and exactly two prediction columns are returned, the second column is used.
nrows	Integer, number of rows in the response-curve grid. If NULL, it is computed automatically.
ncols	Integer, number of columns in the response-curve grid. Defaults to 2 so the response-curve panel stays closer to the map height.
method	Character, the curve type to plot. "profile" uses a single reference profile, "pdp" averages over sampled predictor rows, "ice" draws individual conditional expectation curves, "ice+pdp" overlays the averaged PDP on top of the ICE curves, and "ale" draws accumulated local effects curves.
selected_color	Character, colour used for the clicked-site marker on the map and response curves.
show_selected_ice	Logical, whether to overlay the clicked site's ceteris paribus / ICE curve on the response panels using selected_color. Defaults to TRUE. Ignored when method = "ale".
crosshair	Logical, whether to draw dashed horizontal and vertical guide lines through the selected map cell. Defaults to TRUE.
map_palette	Character vector of colours used to draw the prediction map.
map_title	Character, title shown above the map.
launch	Logical, whether to launch the Shiny app immediately.

Details

The map displays the supplied prediction raster; the model is not refitted inside the app. The curve panel is computed once with the same methods as univariate(). For a fitted prediction function $\hat{f}$ and plotted predictor x_j, "profile" uses a reference-row curve $\hat{f}(z, x_{-j}^{ref})$, "ice" plots sampled row-level curves $\hat{f}(z, x_{-j}^{(i)})$, "pdp" plots their average

$$\hat{f}_{j,PDP}(z) = \frac{1}{m}\sum_{i=1}^{m} \hat{f}(z, x_{-j}^{(i)}),$$

and "ale" plots centred accumulated local effects using the same univariate ALE definitions as univariate(). When a user clicks the map, the clicked cell's covariate value is overlaid on each

panel so the local environmental context can be compared with the fitted response curve and the sampled predictor distribution.

These curves are diagnostic summaries of a fitted prediction model. They do not by themselves establish causal effects, and PDP/ICE/profile curves can evaluate uncommon predictor combinations when predictors are dependent.

Value

A `shiny.appobj` object. If `launch = TRUE`, the app is also run and returned invisibly after it closes.

References

Molnar, C. (2025). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (3rd ed.). https://christophm.github.io/interpretable-ml-book/

Examples

```
if (requireNamespace("terra", quietly = TRUE) &&
    requireNamespace("shiny", quietly = TRUE)) {
  r <- terra::rast(
    ncols = 10,
    nrows = 10,
    nlyrs = 2,
    xmin = 0,
    xmax = 1,
    ymin = 0,
    ymax = 1
  )
  values <- cbind(
    rep(seq(0, 1, length.out = 10), each = 10),
    rep(seq(0, 1, length.out = 10), times = 10)
  )
  terra::values(r) <- values
  names(r) <- c("x1", "x2")

  dat <- terra::as.data.frame(r)
  dat$y <- 1 + 2 * dat$x1 - dat$x2
  fit <- lm(y ~ x1 + x2, data = dat)

  pred_map <- r[[1]]
  terra::values(pred_map) <- dat$y
  names(pred_map) <- "prediction"

  app <- mapcurve(
    fit,
    map = pred_map,
    predictors = r,
    launch = FALSE
  )
  invisible(app)
}
```

`multimodel` *Aggregated response curves across fitted models*

Description

Plot profile, partial dependence, or accumulated local effects curves for several fitted models on a common predictor grid, then combine the model-specific curves into a summary curve. This can be used for ensembles, cross-validation folds, bootstrap refits, or comparisons across different background samples and related training sets.

Usage

```
multimodel(
  models,
  x = NULL,
  predict_data = NULL,
  fun = NULL,
  ...,
  method = c("pdp", "ale", "profile"),
  n = 100,
  background_n = n,
  agg = mean,
  weights = NULL,
  interval = c("sd", "none", "quantile"),
  interval_level = 0.8,
  ylab = "Prediction",
  rug = TRUE,
  ylim = NULL,
  color = "deepskyblue4",
  response = NULL,
  nrows = NULL,
  ncols = NULL,
  show_models = FALSE
)
```

Arguments

`models`	A list of fitted models that support prediction. These may be ensemble members, cross-validation folds, bootstrap refits, or other repeated fits. Models should be fitted to compatible predictor variables and return predictions on the same response scale.
`x`	A data frame or raster containing predictor variables. If `predict_data` is provided, this argument is ignored.
`predict_data`	A data frame containing values at which predictions should be made. If `NULL`, `x` must be provided.

fun	A function used to generate predictions from the model, or a list of functions the same length as models. If NULL, the generic predict() is used for every model.
...	Additional arguments passed to each prediction function. For mixed model types with different prediction interfaces, prefer supplying model-specific wrappers through fun.
method	Character, the curve type to plot. "profile" uses a single reference profile, "pdp" averages over sampled predictor rows before combining ensemble members, and "ale" draws accumulated local effects curves.
n	Integer, number of points to sample for each numeric predictor variable (default: 100). For "ale", n sets the maximum number of intervals used to estimate local effects for numeric predictors.
background_n	Integer, number of randomly sampled background rows used for "pdp" (default: n).
agg	Function used to combine model-specific predictions at each point along the curve. Defaults to mean.
weights	Optional numeric vector of model weights with the same length as models.
interval	Character, interval type drawn around the summary curve. "sd" draws a standard deviation ribbon, "quantile" draws a central quantile ribbon using interval_level, and "none" disables the ribbon.
interval_level	Numeric in (0, 1) giving the central quantile width used when interval = "quantile". Ignored otherwise.
ylab	Character, label for the y-axis (default: "Prediction").
rug	Logical, whether to include a rug plot along the x-axis (default: TRUE).
ylim	Numeric vector of length 2, specifying the limits of the y-axis. If NULL, limits are automatically set.
color	Character, colour of the response curve (default: "deepskyblue4").
response	Optional column name or index to select when fun returns multiple predictions per row. If NULL and exactly two prediction columns are returned, the second column is used.
nrows	Integer, number of rows in the plot grid. If NULL, it is automatically determined.
ncols	Integer, number of columns in the plot grid. If NULL, it is automatically determined.
show_models	Logical, whether to overlay individual model curves beneath the summary curve (default: FALSE).

Details

multimodel() is useful for formal ensemble modelling workflows, such as species distribution models where alternative algorithms or model specifications are fit to the same response and predictor set. It is also useful for repeated-fit comparisons, such as cross-validation folds, bootstrap or bagged refits, models trained with different background samples, or sensitivity checks across related training sets. Each fitted model is first converted to the same one-dimensional curve type used by univariate(). For model r, write this curve as $h_r(z)$. The displayed ensemble curve is

$$H(z) = A\{h_1(z), \ldots, h_R(z)\},$$

where A is `agg`. With the default `agg = mean` and no weights, this is the arithmetic ensemble mean. If `weights` are supplied and `agg` is `mean`, the package uses

$$H(z) = \frac{\sum_{r=1}^{R} w_r h_r(z)}{\sum_{r=1}^{R} w_r}.$$

`method = "profile"` uses a single reference row, $\hat{f}_r(z, x_{-j}^{ref})$; `method = "pdp"` averages each model's predictions over sampled background rows,

$$h_r(z) = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_r(z, x_{-j}^{(i)});$$

and `method = "ale"` accumulates centred local prediction differences using the same univariate ALE definitions as `univariate()`. These definitions follow the model-agnostic PDP and ALE notation summarized by Molnar (2025).

The default `interval = "sd"` draws $H(z) \pm s(z)$, using the ordinary standard deviation across model curves or, with weights,

$$s(z) = \sqrt{\frac{\sum_{r=1}^{R} w_r \{h_r(z) - H(z)\}^2}{\sum_{r=1}^{R} w_r}}.$$

`interval = "quantile"` instead draws central pointwise quantiles of the model-specific curve values.

Value

A `ggplot2` object containing the response curves arranged in a grid.

References

Molnar, C. (2025). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (3rd ed.). https://christophm.github.io/interpretable-ml-book/

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189-1232.

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B*, 82(4), 1059-1086.

Examples

```
if (requireNamespace("mgcv", quietly = TRUE)) {
  data(iris)
  predictors <- iris[, c("Sepal.Width", "Petal.Length")]

  models <- list(
    lm(Sepal.Length ~ Sepal.Width + Petal.Length, data = iris),
    mgcv::gam(
      Sepal.Length ~ s(Sepal.Width) + s(Petal.Length),
      data = iris
```

```
  )
)

response_plot <- multimodel(
  models,
  predictors,
  method = "pdp",
  background_n = 50,
  show_models = TRUE
)
print(response_plot)
}
```

`univariate` *Univariate model-agnostic response curves*

Description

Plot how a fitted model's predictions change across one predictor at a time using single-profile, partial dependence, individual conditional expectation, or accumulated local effects curves.

Usage

```
univariate(
  model,
  x = NULL,
  predict_data = NULL,
  fun = NULL,
  ...,
  n = 100,
  background_n = n,
  interval = c("none", "quantile"),
  interval_level = 0.8,
  rug = TRUE,
  ylim = NULL,
  ylab = NULL,
  color = "deepskyblue4",
  response = NULL,
  nrows = NULL,
  ncols = NULL,
  method = c("pdp", "ice", "ice+pdp", "ale", "profile")
)
```

Arguments

`model`	A fitted model object that supports prediction.
`x`	A data frame or raster containing predictor variables. If `predict_data` is provided, this argument is ignored.

predict_data	A data frame containing values at which predictions should be made. If NULL, x must be provided.
fun	A function used to generate predictions from the model. If NULL, the generic predict() is used.
...	Additional arguments passed to fun.
n	Integer, number of points to sample for each numeric predictor variable (default: 100). For "ale", n sets the maximum number of intervals used to estimate local effects for numeric predictors.
background_n	Integer, number of randomly sampled background rows used for "pdp", "ice", and "ice+pdp" (default: n).
interval	Character, interval type used to draw a PDP ribbon for numeric predictors. Only "quantile" is currently supported and only when method = "pdp". Defaults to "none".
interval_level	Numeric in (0, 1) giving the central quantile width used when interval = "quantile". Ignored otherwise.
rug	Logical, whether to include a rug plot along the x-axis (default: TRUE).
ylim	Numeric vector of length 2, specifying the limits of the y-axis. If NULL, limits are automatically set.
ylab	Optional character label for the y-axis. If NULL, the default is "Prediction" for profile, PDP, and ICE methods, and "Accumulated local effect" for method = "ale".
color	Character, colour of the response curve (default: "deepskyblue4").
response	Optional column name or index to select when fun returns multiple predictions per row. If NULL and exactly two prediction columns are returned, the second column is used.
nrows	Integer, number of rows in the plot grid. If NULL, it is automatically determined.
ncols	Integer, number of columns in the plot grid. If NULL, it is automatically determined.
method	Character, the curve type to plot. "profile" uses a single reference profile, "pdp" averages over sampled predictor rows, "ice" draws individual conditional expectation curves, and "ice+pdp" overlays the averaged PDP on top of the ICE curves. "ale" draws accumulated local effects curves.

Details

Let $\hat{f}$ denote the fitted prediction function, x_j the predictor being plotted, and x_{-j} all other predictors. The "profile" method builds one reference row, using the mean numeric value and modal factor level of each predictor, and plots

$$g_j(z) = \hat{f}(z, x_{-j}^{ref}).$$

This is a single ceteris paribus curve: it is easy to read, but it describes the model only around the chosen reference profile.

For "ice", the curve for sampled row i is

$$g_j^{(i)}(z) = \hat{f}(z, x_{-j}^{(i)}).$$

ICE curves show row-level heterogeneity. `"ice+pdp"` overlays their average. For `"pdp"`, the plotted partial dependence curve is the Monte Carlo average over m sampled background rows,

$$\hat{f}_{j,PDP}(z) = \frac{1}{m}\sum_{i=1}^{m} \hat{f}(z, x_{-j}^{(i)}).$$

This follows the marginal-distribution PDP definition in Molnar (2025). When predictors are strongly dependent, PDP and ICE curves can include feature combinations that are rare or outside the observed data distribution.

For `"ale"`, numeric x_j is split into intervals with breakpoints $z_0 < \cdots < z_K$. For interval $N_j(k) = \{i : x_j^{(i)} \in (z_{k-1}, z_k]\}$, the local effect is estimated by

$$\Delta_{j,k} = \frac{1}{n_j(k)} \sum_{i \in N_j(k)} \left[\hat{f}(z_k, x_{-j}^{(i)}) - \hat{f}(z_{k-1}, x_{-j}^{(i)})\right].$$

The reported value at the interval centre $c_k = (z_{k-1} + z_k)/2$ is accumulated and centred,

$$\hat{f}_{j,ALE}(c_k) = \left(\sum_{\ell=1}^{k} \Delta_{j,\ell} - \frac{1}{2}\Delta_{j,k}\right) - \frac{\sum_{k=1}^{K} n_j(k)\tilde{f}_{j,ALE}(c_k)}{\sum_{k=1}^{K} n_j(k)},$$

where $\tilde{f}_{j,ALE}(c_k)$ is the uncentred accumulated value. ALE averages local prediction differences within observed intervals and then centres the curve so its sample-weighted mean effect is zero.

Value

A `ggplot2` object containing the response curves arranged in a grid.

References

Molnar, C. (2025). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (3rd ed.). https://christophm.github.io/interpretable-ml-book/

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189-1232.

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1), 44-65.

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B*, 82(4), 1059-1086.

Examples

```
data(iris)
model <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, data = iris)
response_plot <- univariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")]
)
print(response_plot)
```

```
pdp_plot <- univariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")],
  method = "pdp",
  n = 25,
  background_n = 50,
  interval = "quantile",
  interval_level = 0.8
)
print(pdp_plot)

ice_plot <- univariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")],
  method = "ice+pdp",
  n = 25,
  background_n = 50
)
print(ice_plot)

ale_plot <- univariate(
  model,
  x = iris[, c("Sepal.Width", "Petal.Length", "Petal.Width")],
  method = "ale",
  n = 20
)
print(ale_plot)
```

Index

bivariate, 2
bivariate(), *6*

interactions, 5
interactions(), *3*

mapcurve, 7
multimodel, 11

univariate, 14
univariate(), *9*, *12*, *13*