Package 'flashlight'

October 13, 2019

Type Package

Title Shed Light on Black Box Machine Learning Models

Version 0.3.0

Date 2019-10-12

Maintainer Michael Mayer <mayermichael79@gmail.com>

Description Shed light on black box machine learning models by the help of model performance, permutation variable importance (Fisher et al. (2018) <arxiv:1801.01489>), ICE profiles, partial dependence (Friedman J. H. (2001) <doi:10.1214/aos/1013203451>), accumulated local effects (Apley D. W. (2016) <arXiv:1612.08468>), further effects plots, and variable contribution breakdown for single observations (Gosiewska and Biecek (2019) <arxiv:1903.11420>). All tools are implemented to work with case weights and allow for stratified analysis. Furthermore, multiple flashlights can be combined and analyzed together.

License GPL (>= 2)

Depends R (>= 3.5.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

Imports stats, utils, dplyr, tidyr, rlang, ggplot2, ggpubr, MetricsWeighted (>= 0.2.0)

Suggests knitr, lubridate, ranger, xgboost, caret, moderndive

RoxygenNote 6.1.1

NeedsCompilation no

Author Michael Mayer [aut, cre, cph]

Repository CRAN

Date/Publication 2019-10-13 07:30:02 UTC

R topics documented:

`ale_profile` *ALE profile*

Description

Internal function used by `light_profile` to calculate ALE profiles.

Usage

```
ale_profile(x, v, breaks = NULL, n_bins = 11, cut_type = c("equal",
  "quantile"), value_name = "value", counts_name = "counts",
  counts = TRUE, counts_weighted = FALSE, pred = NULL,
  evaluate_at = NULL, indices = NULL, n_max = 1000, seed = NULL,
  two_sided = FALSE, calibrate = TRUE)
```

Arguments

x	An object of class `flashlight`.
v	The variable to be profiled.
breaks	Cut breaks for a numeric `v`. Only used if no `evaluate_at` is specified.
n_bins	Maxmium number of unique values to evaluate for numeric `v`. Only used if no `evaluate_at` is specified.
cut_type	For the default "equal", bins of equal width are created for `v` by `pretty`. Choose "quantile" to create quantile bins.
value_name	Column name containing the profile value. Defaults to "value".
counts_name	Name of the column containing counts if `counts` is TRUE.
counts	Should counts be added?
counts_weighted	If `counts` is TRUE: Should counts be weighted by the case weights? If TRUE, the sum of `w` is returned by group.
pred	Optional vector with predictions.
evaluate_at	Vector with values of `v` used to evaluate the profile. Only relevant for type = "partial dependence".
indices	A vector of row numbers to consider.
n_max	Maximum number of ICE profiles to calculate within interval (not within data).
seed	Integer random seed passed to `light_ice`.
two_sided	Standard ALE profiles are calculated via left derivatives. Set to TRUE if two-sided derivatives should be calculated. Only works for continuous `v`. More specifically: Usually, local effects at value x are calculated using points between x-e and x. Set `ale_two_sided = TRUE` to use points between x-e/2 and x+e/2.
calibrate	Should values be calibrated based on average preditions? Default is TRUE.

Value

A tibble containing results.

all_identical *all_identical*

Description

Checks if an aspect is identical for all elements in a nested list. The aspect is specified by `fun`, e.g. '[[', followed by the element name to compare.

Usage

```
all_identical(x, fun, ...)
```

Arguments

`x`	A nested list of objects.
`fun`	Function used to extract information of each element of `x`.
`...`	Further arguments passed to `fun`.

Value

A logical vector of length one.

Examples

```
x <- list(a = 1, b = 2)
y <- list(a = 1, b = 3)
all_identical(list(x, y), `[[`, "a")
all_identical(list(x, y), `[[`, "b")
```

auto_cut *Discretizes a Vector*

Description

This function takes a vector `x` and returns a list with information on disretized version of `x`, see `return` for details on the resulting object.

Usage

```
auto_cut(x, breaks = NULL, n_bins = 27, cut_type = c("equal",
  "quantile"), x_name = "value", level_name = "level", ...)
```

Arguments

x	A vector.
breaks	An optional vector of breaks. Only relevant for numeric x.
n_bins	If x is numeric and no breaks are provided, this is the maximum number of bins allowed or to be created (approximately).
cut_type	For the default type "equal", bins of equal width are created by pretty. Choose "quantile" to create quantile bins.
x_name	Column name with the values of x in the output.
level_name	Column name with the bin labels of x in the output.
...	Further arguments passed to cut3.

Details

The construction of level names can be controlled by passing ... arguments to formatC.

Value

A list with the following four elements:

- data A data.frame with colums x_name and level_name each with the same length as x. The column x_name has values in output bin_means while the column level_name has values in bin_labels.
- breaks A vector of increasing and unique breaks used to cut a numeric x with too many distinct levels. NULL otherwise.
- bin_means The midpoints of subsequent breaks, or if there are no breaks in the output, factor levels or distinct values of x.
- bin_labels Break labels of the form "(low, high]" if there are breaks in the output, otherwise the same as bin_means. Same order as bin_means.

Examples

```
auto_cut(1:10, n_bins = 3)
auto_cut(c(NA, 1:10), n_bins = 3)
auto_cut(1:10, breaks = 3:4, n_bins = 3)
auto_cut(1:10, n_bins = 3, cut_type = "quantile")
auto_cut(LETTERS[4:1], n_bins = 2)
auto_cut(factor(LETTERS[1:4], LETTERS[4:1]), n_bins = 2)
auto_cut(990:1100, n_bins = 3, big.mark = "'", format = "fg")
auto_cut(c(0.0001, 0.0002, 0.0003, 0.005), n_bins = 3, format = "fg")
```

cut3 *Modified cut*

Description

Slightly modified version of base::cut.default. Both modifications refer to the construction of break labels. Firstly, ... arguments are passed to `formatC` in formatting the numbers in the labels. Secondly, a separator between the two numbers can be specified with default ", ".

Usage

```
cut3(x, breaks, labels = NULL, include.lowest = FALSE, right = TRUE,
  dig.lab = 3L, ordered_result = FALSE, sep = ", ", ...)
```

Arguments

`x`	Numeric vector.
`breaks`	Numeric vector of cut points or a single number specifying the number of intervals desired.
`labels`	Labels for the levels of the final categories.
`include.lowest`	Flag if minimum value should be added to intervals of type (,] (or maximum for [,)).
`right`	Flag if intervals should be closed to the right or left.
`dig.lab`	Number of significant digits passed to formatC.
`ordered_result`	Flag if resulting output vector should be ordered.
`sep`	Separater between from-to labels.
`...`	Arguments passed to `formatC`.

Value

Vector of the same length as x.

Examples

```
x <- 998:1001
cut3(x, breaks = 2)
cut3(x, breaks = 2, big.mark = "'", sep = ":")
```

`flashlight` *Create or Update a flashlight*

Description

Creates or updates a `flashlight` object. If a flashlight is to be created, all arguments are optional except `label`. If a flashlight is to be updated, all arguments are optional up to `x` (the flashlight to be updated).

Usage

```
flashlight(x, ...)

## Default S3 method:
flashlight(x, model = NULL, data = NULL, y = NULL,
  predict_function = predict, linkinv = function(z) z, w = NULL,
  by = NULL, metrics = list(rmse = rmse), label = NULL, ...)

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

Arguments

`x`	An object of class `flashlight`. If not provided, a new flashlight is created based on further input. Otherwise, `x` is updated based on further input.
`...`	Arguments passed from or to other functions.
`model`	A fitted model of any type. Most models require a customized `predict_function`.
`data`	A `data.frame` or `tibble` used as basis for calculations.
`y`	Variable name of response.
`predict_function`	A real valued function with two arguments: A model and a data of the same structure as `data`. Only the order of the two arguments matter, not their names.
`linkinv`	An inverse transformation function applied after `predict_function`.
`w`	A variable name of case weights.
`by`	A character vector with names of grouping variables.
`metrics`	A named list of metrics. Here, a metric is a function with exactly four arguments: actual, predicted, w (case weights) and `...` like those in package `MetricsWeighted`.
`label`	Name of the flashlight. Required.

Value

An object of class `flashlight` (and `list`) containing each input (except `x`) as element.

Methods (by class)

- `default`: Used to create a flashlight object. No `x` has to be passed in this case.
- `flashlight`: Used to update an existing flashlight object.

See Also

`multiflashlight`.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
(fl <- flashlight(model = fit, data = iris, y = "Sepal.Length", label = "ols"))
(fl_updated <- flashlight(fl, linkinv = exp))
```

---

grouped_stats *Grouped Weighted Means or Quartiles*

Description

Calculates weighted means or quartiles (and counts) of a variable grouped by optional columns.

Usage

```
grouped_stats(data, x, w = NULL, by = NULL, stats = c("mean",
  "quartiles"), counts = TRUE, counts_weighted = FALSE,
  counts_name = "counts", value_name = x, q1_name = "q1",
  q3_name = "q3", ...)
```

Arguments

`data`	A `data.frame`.
`x`	Variable name in `data` to summarize.
`w`	Optional name of the column in `data` with case weights.
`by`	An optional vector of column names in `data` used to group the results.
`stats`	Statistic to calculate: "mean" or "quartiles".
`counts`	Should group counts be added?
`counts_weighted`	Should counts be weighted by the case weights? If TRUE, the sum of `w` is returned by group.
`counts_name`	Name of column in the resulting `data.frame` containing the counts.
`value_name`	Name of the resulting column with mean or median.
`q1_name`	Name of the resulting column with first quartile values. Only relevant for `stats` "quartiles".
`q3_name`	Name of the resulting column with third quartile values. Only relevant for `stats` "quartiles".
`...`	Additional arguments passed to `MetricsWeighted::weighted_mean` or `weighted_quartiles`.

Value

A data.frame with columns by, x and optionally counts_name.

Examples

```
grouped_stats(iris, "Sepal.Width")
grouped_stats(iris, "Sepal.Width", stats = "quartiles")
grouped_stats(iris, "Sepal.Width", w = "Petal.Width")
grouped_stats(iris, "Sepal.Width", w = "Petal.Width", counts_weighted = TRUE)

grouped_stats(iris, "Sepal.Width", by = "Species")
grouped_stats(iris, "Sepal.Width", stats = "quartiles", by = "Species")
grouped_stats(iris, "Sepal.Width", w = "Petal.Width", by = "Species")
grouped_stats(iris, "Sepal.Width", w = "Petal.Width",
  counts_weighted = TRUE, by = "Species")

grouped_stats(iris, "Sepal.Width", counts = FALSE)
grouped_stats(iris, "Sepal.Width", counts_name = "n",
  stats = "quartiles", q1_name = "p25", q3_name = "p75")
```

---

is.flashlight *Check functions for flashlight Classes*

Description

Checks if an object inherits specific class relevant for the flashlight package.

Usage

```
is.flashlight(x)

is.multiflashlight(x)

is.light(x)

is.light_performance(x)

is.light_performance_multi(x)

is.light_importance(x)

is.light_importance_multi(x)

is.light_breakdown(x)

is.light_breakdown_multi(x)

is.light_ice(x)
```

```
is.light_ice_multi(x)

is.light_profile(x)

is.light_profile_multi(x)

is.light_effects(x)

is.light_effects_multi(x)
```

Arguments

x	Any object.

Value

A logical vector of length one.

Functions

- `is.multiflashlight`: Check for multiflashlight object.
- `is.light`: Check for light object.
- `is.light_performance`: Check for light_performance object.
- `is.light_performance_multi`: Check for light_performance_multi object.
- `is.light_importance`: Check for light_importance object.
- `is.light_importance_multi`: Check for light_importance_multi object.
- `is.light_breakdown`: Check for light_breakdown object.
- `is.light_breakdown_multi`: Check for light_breakdown_multi object.
- `is.light_ice`: Check for light_ice object.
- `is.light_ice_multi`: Check for light_ice_multi object.
- `is.light_profile`: Check for light_profile object.
- `is.light_profile_multi`: Check for light_profile_multi object.
- `is.light_effects`: Check for light_effects object.
- `is.light_effects_multi`: Check for light_effects_multi object.

Examples

```
a <- flashlight(label = "a")
is.flashlight(a)
is.flashlight("a")
```

light_breakdown *Variable Contribution Breakdown for Single Observation*

Description

Calculates sequential additive variable contribution to the prediction of a single observation, see Gosiewska and Biecek [1] and the details below.

Usage

```
light_breakdown(x, ...)

## Default S3 method:
light_breakdown(x, ...)

## S3 method for class 'flashlight'
light_breakdown(x, new_obs, data = x$data,
  by = x$by, v = NULL, order_by_importance = TRUE, top_m = Inf,
  n_max = Inf, seed = NULL, use_linkinv = FALSE,
  after_name = "after", before_name = "before", label_name = "label",
  variable_name = "variable", step_name = "step",
  description_name = "description", digits = 3, ...)

## S3 method for class 'multiflashlight'
light_breakdown(x, ...)
```

Arguments

x	An object of class flashlight or multiflashlight.
...	Further arguments passed to prettyNum to format numbers in description text.
new_obs	One single new observation to calculate variable attribution for. Needs to be a data.frame of same structure as data.
data	An optional data.frame.
by	An optional vector of column names used to filter data for rows with equal values in "by" variables as new_obs.
v	Vector of variables to assess contribution for. Defaults to all except those specified by "y", "w" and "by".
order_by_importance	Logical flag indicated if v should be ordered automatically based on individual contribution (see description).
top_m	Maximum number of steps/variable contributions to be calculated.
n_max	Maximum number of rows in data to consider. Set to lower value if data is large.
seed	An integer random seed used to shuffle rows if n_max is smaller than the number of rows in data.

use_linkinv	Should retransformation function be applied? Default is FALSE.
after_name	Column name in resulting data containing prediction after the step in step_name. Defaults to "after".
before_name	Column name in resulting data containing prediction before the step in step_name. Defaults to "before".
label_name	Column name in resulting data containing the label of the flashlight. Defaults to "label".
variable_name	Column name in resulting data containing the variable names. Defaults to "variable".
step_name	Column name in resulting data containing the step of the prediction process. Defaults to "step".
description_name	Column name in resulting data containing the description text to be visualized. Defaults to "description".
digits	Passed to prettyNum to format numbers in description text.

Details

The breakdown algorithm works as follows: First, the visit order (x_1, ..., x_m) of the variables v is specified. Then, in the query data, the column x_1 is set to the value of x_1 of the single observation new_obs to be explained. The change in the (weighted) average prediction on data measures the contribution of x_1 on the prediction of new_obs. This procedure is iterated over all x_i until eventually, all rows in data are identical to new_obs. A complication with this approach is that the visit order is relevant, at least for non-additive models. Ideally, the algorithm could be repeated for all possible permutations of v and its results averaged per variable. This is basically what SHAP values do, see [1] for an explanation. Unfortunately, there is no efficient way to do this in a model agnostic way. Thus we use the short-cut described in [1]. The variables are sorted by the size of their contribution in the same way as the breakdown algorithm but without iteration, i.e. starting from the original query data for each variable x_i. Note that the minimum required elements in the (multi-) flashlight are "y", "predict_function", "model", and "data". The latter can also directly be passed to light_breakdown. Note that by default, no retransformation function is applied.

Value

An object of class light_breakdown, light (and a list) with the following elements.

- data A tibble with results. Can be used to build fully customized visualizations.
- by Same as input by.
- after_name Same as input after_name.
- before_name Same as input before_name.
- label_name Same as input label_name.
- variable_name Same as input variable_name.
- step_name Same as input step_name.
- description_name Same as input description_name.

Methods (by class)

- `default`: Default method not implemented yet.
- `flashlight`: Variable attribution to single observation for a flashlight.
- `multiflashlight`: Variable attribution to single observation for a multiflashlight.

References

[1] A. Gosiewska and P. Biecek (2019). IBREAKDOWN: Uncertainty of model explanations for non-additive predictive models. ArXiv <arxiv.org/abs/1903.11420>.

See Also

`plot.light_breakdown`.

Examples

```
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
fit_full <- lm(Sepal.Length ~ ., data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part), by = "Species")
light_breakdown(mod_full, new_obs = iris[1, ])
light_breakdown(mods, new_obs = iris[1, ])
light_breakdown(mods, new_obs = iris[1, ], top_m = 2)

ir <- iris
ir$log_sl <- log(ir$Sepal.Length)
fit_lm <- lm(log_sl ~ Petal.Length, data = ir)
fit_glm <- glm(Sepal.Length ~ Petal.Length, data = ir, family = Gamma(link = log))
fl_lm <- flashlight(model = fit_lm, label = "lm", y = "log_sl", linkinv = exp)
fl_glm <- flashlight(model = fit_glm, label = "glm", y = "Sepal.Length",
  predict_function = function(m, X) predict(m, X, type = "response"))
fls <- multiflashlight(list(fl_lm, fl_glm), data = ir)
light_breakdown(fls, new_obs = ir[1, ])$data
light_breakdown(fls, new_obs = ir[1, ], use_linkinv = TRUE)$data
```

`light_check` *Check flashlight*

Description

Checks if an object of class `flashlight` or `multiflashlight` is consistently defined.

Usage

```
light_check(x)

## Default S3 method:
light_check(x)

## S3 method for class 'flashlight'
light_check(x)

## S3 method for class 'multiflashlight'
light_check(x)
```

Arguments

x	An object of class `flashlight` or `multiflashlight`.

Value

The input `x` or an error message.

Methods (by class)

- `default`: Default check method not implemented yet.
- `flashlight`: Checks if a flashlight object is consistently defined.
- `multiflashlight`: Checks if a multiflashlight object is consistently defined.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
fit_log <- lm(log(Sepal.Length) ~ ., data = iris)
fl <- flashlight(fit, data = iris, y = "Sepal.Length", label = "ols")
fl_log <- flashlight(fit_log,  y = "Sepal.Length", label = "ols", linkinv = exp)
light_check(fl)
light_check(fl_log)
```

`light_combine` *Combine Objects*

Description

Combines a list of similar objects each of class `light` by row binding `data.frame` slots and retaining the other slots from the first list element.

Usage

```
light_combine(x, ...)

## Default S3 method:
light_combine(x, ...)

## S3 method for class 'light'
light_combine(x, new_class = NULL, ...)

## S3 method for class 'list'
light_combine(x, new_class = NULL, ...)
```

Arguments

x	A list of objects of the same class.
...	Further arguments passed from or to other methods.
new_class	An optional vector with additional class names to be added to the output.

Value

If x is a list, an object like each element but with unioned rows in data slots.

Methods (by class)

- default: Default method not implemented yet.
- light: Since there is nothing to combine, the input is returned except for additional classes.
- list: Combine a list of similar light objects.

Examples

```
fit_lm <- lm(Sepal.Length ~ ., data = iris)
fit_glm <- glm(Sepal.Length ~ ., family = Gamma(link = "log"), data = iris)
mod_lm <- flashlight(model = fit_lm, label = "lm", data = iris, y = "Sepal.Length")
mod_glm <- flashlight(model = fit_glm, label = "glm", data = iris, y = "Sepal.Length",
                      predict_function = function(object, newdata)
                      predict(object, newdata, type = "response"))
mods <- multiflashlight(list(mod_lm, mod_glm))
perf_lm <- light_performance(mod_lm)
perf_glm <- light_performance(mod_glm)
manual_comb <- light_combine(list(perf_lm, perf_glm),
  new_class = "light_performance_multi")
auto_comb <- light_performance(mods)
all.equal(manual_comb, auto_comb)
light_combine(perf_lm)
```

light_effects *Combination of Response, Predicted, Partial Dependence, and ALE Profiles*

Description

Calculates response- prediction-, partial dependence, and ALE profiles of a (multi-)flashlight with respect to a covariable v.

Usage

```
light_effects(x, ...)

## Default S3 method:
light_effects(x, ...)

## S3 method for class 'flashlight'
light_effects(x, v, data = NULL, by = x$by,
  stats = c("mean", "quartiles"), breaks = NULL, n_bins = 11,
  cut_type = c("equal", "quantile"), use_linkinv = TRUE,
  value_name = "value", q1_name = "q1", q3_name = "q3",
  label_name = "label", type_name = "type", counts_name = "counts",
  counts_weighted = FALSE, v_labels = TRUE, pred = NULL,
  pd_indices = NULL, pd_n_max = 1000, pd_seed = NULL,
  ale_two_sided = TRUE, ...)

## S3 method for class 'multiflashlight'
light_effects(x, v, data = NULL,
  breaks = NULL, n_bins = 11, cut_type = c("equal", "quantile"), ...)
```

Arguments

x	An object of class flashlight or multiflashlight.
...	Further arguments passed to cut3 resp. formatC in forming the cut breaks of the v variable.
v	The variable to be profiled.
data	An optional data.frame.
by	An optional vector of column names used to additionally group the results.
stats	Statistic to calculate for the response profile: "mean" or "quartiles".
breaks	Cut breaks for a numeric v.
n_bins	Maxmium number of unique values to evaluate for numeric v.
cut_type	For the default "equal", bins of equal width are created for v by pretty. Choose "quantile" to create quantile bins (recommended if interested in ALE).
use_linkinv	Should retransformation function be applied? Default is TRUE.

value_name	Column name in resulting data objects containing the profile value. Defaults to "value".
q1_name	Name of the resulting column with first quartile values. Only relevant for stats "quartiles".
q3_name	Name of the resulting column with third quartile values. Only relevant for stats "quartiles".
label_name	Column name in resulting data containing the label of the flashlight. Defaults to "label".
type_name	Name of the column in data containing type.
counts_name	Name of the column containing counts.
counts_weighted	Should counts be weighted by the case weights? If TRUE, the sum of w is returned by group.
v_labels	If FALSE, return group centers of v instead of labels. Only relevant if v is numeric with many distinct values. In that case useful if e.g. different flashlights use different data sets.
pred	Optional vector with predictions (after application of inverse link). Can be used to avoid recalculation of predictions over and over if the functions is to be repeatedly called for different v and predictions are computationally expensive to make.
pd_indices	A vector of row numbers to consider in calculating partial dependence and ALE profiles. Useful to force all flashlights to use the same basis for calculations of partial dependence and ALE.
pd_n_max	Maximum number of ICE profiles to consider for partial depencence and ALE calculation (will be randomly picked from data).
pd_seed	An integer random seed used to sample ICE profiles for partial dependence and ALE.
ale_two_sided	If TRUE, v is continuous and breaks are passed or being calculated, then two-sided derivatives are calculated for ALE instead of left derivatives. This aligns the results better with the x labels. More specifically: Usually, local effects at value x are calculated using points between x-e and x. Set ale_two_sided = TRUE to use points between x-e/2 and x+e/2.

Details

Note that ALE profiles are being calibrated by (weighted) average predictions. The resulting level might be quite different from the one of the partial dependence profiles.

Value

An object of classes light_effects, light (and a list) with the following elements.

- response A tibble containing the response profiles.
- predicted A tibble containing the prediction profiles.
- pd A tibble containing the partial dependence profiles.

- by Same as input by.
- v The variable(s) evaluated.
- stats Same as input stats.
- value_name Same as input value_name.
- q1_name Same as input q1_name.
- q3_name Same as input q3_name.
- label_name Same as input label_name.
- type_name Same as input type.
- counts_name Same as input counts_name.

Methods (by class)

- default: Default method.
- flashlight: Profiles for a flashlight object.
- multiflashlight: Effect profiles for a multiflashlight object.

See Also

light_profile, plot.light_effects.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))

light_effects(mod_full, v = "Species")
light_effects(mod_full, v = "Species", stats = "quartiles")

light_effects(mod_full, v = "Petal.Width")
light_effects(mod_full, v = "Petal.Width", v_label = FALSE)
light_effects(mod_full, v = "Petal.Width", stats = "quartiles")
light_effects(mod_full, v = "Petal.Width", n_bins = 3)
light_effects(mod_full, v = "Petal.Width", n_bins = 3, format = "f")
light_effects(mod_full, v = "Petal.Width", breaks = 0:3)

light_effects(mod_full, v = "Petal.Width", by = "Species")

light_effects(mods, v = "Petal.Width")
light_effects(mods, v = "Petal.Width", by = "Species")
light_effects(mods, v = "Petal.Width", by = "Species", stats = "quartiles")
```

light_ice *Individual Conditional Expectation (ICE)*

Description

Generates Individual Conditional Expectation (ICE) profiles. An ICE profile shows how the prediction of an observation changes if one or multiple variables are systematically changed across its ranges, holding all other values fixed [1]. The curves can be centered in order to increase visibility of interaction effects. Centering is done within subgroups specified by "by".

Usage

```
light_ice(x, ...)

## Default S3 method:
light_ice(x, ...)

## S3 method for class 'flashlight'
light_ice(x, v = NULL, data = x$data, by = x$by,
  evaluate_at = NULL, breaks = NULL, grid = NULL, n_bins = 27,
  cut_type = c("equal", "quantile"), indices = NULL, n_max = 20,
  seed = NULL, use_linkinv = TRUE, center = FALSE,
  value_name = "value", label_name = "label", id_name = "id", ...)

## S3 method for class 'multiflashlight'
light_ice(x, ...)
```

Arguments

x	An object of class `flashlight` or `multiflashlight`.
...	Further arguments passed to or from other methods.
v	The variable to be profiled.
data	An optional `data.frame`.
by	An optional vector of column names used to additionally group the results.
evaluate_at	Vector with values of `v` used to evaluate the profile.
breaks	Instead of `evaluate_at` (and `grid`), cut points for `x` can be provided. From them, `evaluate_at` values are calculates as averages.
grid	A `data.frame` with grid values as those generated by `expand.grid`.
n_bins	Maximum number of unique values to evaluate for numeric `v`. Only used in neither `grid` nor `evaluate_at` is specified.
cut_type	For the default "equal", bins of equal width are created for `v` by `pretty`. Choose "quantile" to create quantile bins. Only used in neither `grid` nor `evaluate_at` is specified.
indices	A vector of row numbers to consider.

n_max	If indices is not given, maximum number of rows to consider. Will be randomly picked from data if necessary.
seed	An integer random seed.
use_linkinv	Should retransformation function be applied? Default is TRUE.
center	Should curves be centered? Default is FALSE. Note that centering will be done at the first evaluation point and within "by" group. It will work also for a grid with multiple columns.
value_name	Column name in resulting data containing the profile value. Defaults to "value".
label_name	Column name in resulting data containing the label of the flashlight. Defaults to "label".
id_name	Column name in resulting data containing the row id of the profile. Defaults to "id_name".

Details

There are two ways to specify the variable(s) to be profiled. The first option is to pass the variable name via v and an optional vector with evaluation points evaluate_at (or breaks). This works for dependence on a single variable. The second option is much more general: You can specify any grid as a data.frame with one or more columns. It can e.g. be generated by a call to expand.grid. Currently, there is no option to pass more than one variable name without such grid. The minimum required elements in the (multi-)flashlight are "predict_function", "model", "linkinv" and "data", where the latest can be passed on the fly. Which rows in data are profiled? This is specified by indices. If not given and n_max is smaller than the number of rows in data, then row indices will be sampled randomly from data. If the same rows should be used for all flashlights in a multiflashlight, there are two options: Either pass a seed (with potentially undesired consequences for subsequent code) or a vector of indices used to select rows. In both cases, data should be the same for all flashlights considered.

Value

An object of class light_ice, light (and a list) with the following elements.

- data A tibble containing the results. Can be used to build fully customized visualizations. Its column names are specified by all other items in this list.
- by Same as input by.
- v The variable(s) evaluated. @item center Flag if ICE curves are centered.
- value_name Same as input value_name.
- label_name Same as input label_name.
- id_name Same as input id_name.

Methods (by class)

- default: Default method not implemented yet.
- flashlight: ICE profiles for a flashlight object.
- multiflashlight: ICE profiles for a multiflashlight object.

References

[1] Goldstein, A. et al. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. Journal of Computational and Graphical Statistics, 24:1 <doi.org/10.1080/10618600.2014.907095>.

See Also

light_profile, plot.light_ice.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))
grid <- expand.grid(Species = levels(iris$Species), Petal.Length = 2:4)
light_ice(mod_full, v = "Species")
light_ice(mod_full, v = "Species", indices = (1:15) * 10)
light_ice(mod_full, v = "Species", evaluate_at = levels(iris$Species))
light_ice(mod_full, grid = grid, data = iris[1,])$data
light_ice(mods, v = "Species", indices = (1:15) * 10)
light_ice(mods, v = "Species", indices = (1:15) * 10, center = TRUE)
light_ice(mods, v = "Petal.Width", n_bins = 5)
light_ice(mods, v = "Petal.Width", by = "Species", n_bins = 5)
light_ice(mods, v = "Petal.Width", by = "Species",
  id_name = "profile", value_name = "val", label_name = "lab")
```

---

light_importance *Permutation Importance*

Description

Calculates performance per variable with respect to a performance measure before and after permuting its values. The difference is a measure of importance, see Fisher et al. 2018 [1].

Usage

```
light_importance(x, ...)

## Default S3 method:
light_importance(x, ...)

## S3 method for class 'flashlight'
light_importance(x, data = x$data, by = x$by,
  metric = x$metrics[1], v = NULL, n_max = Inf, seed = NULL,
  lower_is_better = TRUE, use_linkinv = FALSE,
  metric_name = "metric", value_name = "value", label_name = "label",
```

```
  variable_name = "variable", ...)

## S3 method for class 'multiflashlight'
light_importance(x, ...)
```

Arguments

x	An object of class flashlight or multiflashlight.
...	Further arguments passed to light_performance.
data	An optional data.frame.
by	An optional vector of column names used to additionally group the results.
metric	An optional named list of length one with a metric as element. Defaults to the first metric in the flashlight. The metric needs to be a function with at least four arguments: actual, predicted, case weights w and ....
v	Vector of variables to assess importance for. Defaults to all variables in data.
n_max	Maximum number of rows to consider. Use if data is large.
seed	An integer random seed used to select and shuffle rows.
lower_is_better	Logical flag indicating if lower values in the metric are better or not. If set to FALSE, the increase in metric is multiplied by -1.
use_linkinv	Should retransformation function be applied? Default is FALSE.
metric_name	Name of the resulting column containing the name of the metric. Defaults to "metric".
value_name	Column name in resulting data containing the drop in performance. Defaults to "value".
label_name	Column name in resulting data containing the label of the flashlight. Defaults to "label".
variable_name	Column name in resulting data containing the variable names. Defaults to "variable".

Details

The minimum required elements in the (multi-) flashlight are "y", "predict_function", "model", "data" and "metrics". The latter two can also directly be passed to light_importance. Note that by default, no retransformation function is applied.

Value

An object of class light_importance, light (and a list) with the following elements.

- data A tibble with results. Can be used to build fully customized visualizations. The columns "value_original" and "value_shuffled" provide the performance before and after shuffling.
- by Same as input by.
- metric_name Column name representing the name of the metric. For information only.
- value_name Same as input value_name.
- label_name Same as input label_name.
- variable_name Same as input variable_name.

Methods (by class)

- `default`: Default method not implemented yet.
- `flashlight`: Variable importance for a flashlight.
- `multiflashlight`: Variable importance for a multiflashlight.

References

[1] Fisher A., Rudin C., Dominici F. (2018). All Models are Wrong but many are Useful: Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance. ArXiv <arxiv.org/abs/1801.01489>.

See Also

`most_important`, `plot.light_importance`.

Examples

```
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
fit_full <- lm(Sepal.Length ~ ., data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part), by = "Species")
light_importance(mod_full)
light_importance(mods)

ir <- iris
ir$log_sl <- log(ir$Sepal.Length)
fit_lm <- lm(log_sl ~ Petal.Length, data = ir)
fit_glm <- glm(Sepal.Length ~ Petal.Length, data = ir, family = Gamma(link = log))
fl_lm <- flashlight(model = fit_lm, label = "lm", y = "log_sl", linkinv = exp)
fl_glm <- flashlight(model = fit_glm, label = "glm", y = "Sepal.Length",
  predict_function = function(m, X) predict(m, X, type = "response"))
fls <- multiflashlight(list(fl_lm, fl_glm), data = ir)
light_importance(fls, v = "Petal.Length", seed = 45)
light_importance(fls, v = "Petal.Length", seed = 45, use_linkinv = TRUE)
```

`light_performance` *Model Performance of Flashlight*

Description

Calculates performance of a flashlight with respect to one or more performance measure.

Usage

```
light_performance(x, ...)

## Default S3 method:
light_performance(x, ...)

## S3 method for class 'flashlight'
light_performance(x, data = x$data, by = x$by,
  metrics = x$metrics, use_linkinv = FALSE, metric_name = "metric",
  value_name = "value", label_name = "label", ...)

## S3 method for class 'multiflashlight'
light_performance(x, ...)
```

Arguments

x	An object of class flashlight or multiflashlight.
...	Arguments passed from or to other functions.
data	An optional data.frame.
by	An optional vector of column names used to additionally group the results. Will overwrite x$by.
metrics	An optional named list with metrics. Each metric takes at least four arguments: actual, predicted, case weights w and ....
use_linkinv	Should retransformation function be applied? Default is FALSE.
metric_name	Column name in resulting data containing the name of the metric. Defaults to "metric".
value_name	Column name in resulting data containing the value of the metric. Defaults to "value".
label_name	Column name in resulting data containing the label of the flashlight. Defaults to "label".

Details

The minimal required elements in the (multi-) flashlight are "y", "predict_function", "model", "data" and "metrics". The latter two can also directly be passed to light_performance. Note that by default, no retransformation function is applied.

Value

An object of class light_performance, light (and a list) with the following elements.

- data A tibble containing the results. Can be used to build fully customized visualizations.
- by Same as input by.
- metric_name Same as input metric_name.
- value_name Same as input value_name.
- label_name Same as input label_name.

Methods (by class)

- `default`: Default method not implemented yet.
- `flashlight`: Model performance of flashlight object.
- `multiflashlight`: Model performance of multiflashlight object.

See Also

`plot.light_performance`.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part), by = "Species")
light_performance(mod_full)
light_performance(mod_full, metric_name = "perf",
  value_name = "rmse", label_name = "model")
light_performance(mod_full, by = "Species")
light_performance(mods, by = "Species")
```

---

light_profile *Partial Dependence and other Profiles*

Description

Calculates different types of profiles across covariable values. By default, partial dependence profiles [1] are calculated. Other options are profiles of ALE (accumulated local effects, see [2]), response, predicted values ("M plots" or "marginal plots", see [2]) and residuals. The results are aggregated either by (weighted) means or by (weighted) quartiles. Note that ALE profiles are calibrated by (weighted) average predictions. In contrast to the suggestions in [2], we calculate ALE profiles of factors in the same order as the factor levels. They are not being reordered based on similiarity of other variables.

Usage

```
light_profile(x, ...)

## Default S3 method:
light_profile(x, ...)

## S3 method for class 'flashlight'
light_profile(x, v = NULL, data = NULL,
  by = x$by, type = c("partial dependence", "ale", "predicted",
  "response", "residual"), stats = c("mean", "quartiles"),
  breaks = NULL, n_bins = 11, cut_type = c("equal", "quantile"),
```

```
  use_linkinv = TRUE, value_name = "value", q1_name = "q1",
  q3_name = "q3", label_name = "label", type_name = "type",
  counts_name = "counts", counts = TRUE, counts_weighted = FALSE,
  v_labels = TRUE, pred = NULL, pd_evaluate_at = NULL,
  pd_grid = NULL, pd_indices = NULL, pd_n_max = 1000,
  pd_seed = NULL, pd_center = FALSE, ale_two_sided = FALSE, ...)

## S3 method for class 'multiflashlight'
light_profile(x, v = NULL, data = NULL,
  breaks = NULL, n_bins = 11, cut_type = c("equal", "quantile"),
  pd_evaluate_at = NULL, pd_grid = NULL, ...)
```

Arguments

x	An object of class `flashlight` or `multiflashlight`.
...	Further arguments passed to `cut3` resp. `formatC` in forming the cut breaks of the `v` variable. Not relevant for partial dependence and ALE profiles.
v	The variable to be profiled.
data	An optional `data.frame`.
by	An optional vector of column names used to additionally group the results.
type	Type of the profile: Either "partial dependence", "ale", "predicted", "response", or "residual".
stats	Statistic to calculate: "mean" or "quartiles". For ALE profiles, only "mean" makes sense.
breaks	Cut breaks for a numeric `v`.
n_bins	Maxmium number of unique values to evaluate for numeric `v`. Only used if neither `grid` nor `pd_evaluate_at` is specified.
cut_type	For the default "equal", bins of equal width are created for `v` by `pretty`. Choose "quantile" to create quantile bins.
use_linkinv	Should retransformation function be applied? Default is TRUE.
value_name	Column name in resulting `data` containing the profile value. Defaults to "value".
q1_name	Name of the resulting column with first quartile values. Only relevant for `stats` "quartiles".
q3_name	Name of the resulting column with third quartile values. Only relevant for `stats` "quartiles".
label_name	Column name in resulting `data` containing the label of the flashlight. Defaults to "label".
type_name	Column name in the resulting `data` with the plot `type`.
counts_name	Name of the column containing counts if `counts` is TRUE.
counts	Should counts be added?
counts_weighted	If `counts` is TRUE: Should counts be weighted by the case weights? If TRUE, the sum of `w` is returned by group.

v_labels	If FALSE, return group centers of v instead of labels. Only relevant for types "response", "predicted" or "residual" and if v is being binned. In that case useful if e.g. different flashlights use different data sets and bin labels would not match.
pred	Optional vector with predictions (after application of inverse link). Can be used to avoid recalculation of predictions over and over if the functions is to be repeatedly called for different v and predictions are computationally expensive to make. Only relevant for type = "predicted" and type = "ale".
pd_evaluate_at	Vector with values of v used to evaluate the profile. Only relevant for type = "partial dependence" and "ale".
pd_grid	A data.frame with grid values, e.g. generated by expand.grid. Only used for type = "partial dependence".
pd_indices	A vector of row numbers to consider in calculating partial dependence profiles. Only used for type = "partial dependence" and "ale".
pd_n_max	Maximum number of ICE profiles to calculate (will be randomly picked from data). Only used for type = "partial dependence" and "ale".
pd_seed	Integer random seed used to select ICE profiles. Only used for type = "partial dependence" and "ale".
pd_center	Should ICE curves be centered within by subsets before caclulating partial dependence profiles? This option is interesting together with stats = "quartiles" in order to visualize interaction strength.
ale_two_sided	If TRUE, v is continuous and breaks are passed or being calculated, then two-sided derivatives are calculated for ALE instead of left derivatives. More specifically: Usually, local effects at value x are calculated using points between x-e and x. Set ale_two_sided = TRUE to use points between x-e/2 and x+e/2.

Details

For numeric covariables v with more than n_bins disjoint values, its values are binned. Alternatively, breaks can be provided to specify the binning. For partial dependence profiles (and partly also ALE profiles), this behaviour can be overritten either by providing a vector of evaluation points (pd_evaluate_at) or an evaluation pd_grid. By the latter we mean a data frame with column name(s) with a (multi-)variate evaluation grid. For partial dependence, ALE and prediction profiles, "model", "predict_function", linkinv" and "data" are required. For response profiles its just "y", "linkinv" and "data". "data" can be passed on the fly for both types.

Value

An object of classes light_profile, light (and a list) with the following elements.

- data A tibble containing results. Can be used to build fully customized visualizations. Its column names are specified by all other items in this list.
- by Names of group by variable.
- v The variable(s) evaluated.
- type Same as input type. For information only.
- stats Same as input stats.

- value_name Same as input value_name.
- q1_name Same as input q1_name.
- q3_name Same as input q3_name.
- label_name Same as input label_name.
- type_name Same as input type_name.
- counts_name Same as input counts_name.

Methods (by class)

- default: Default method not implemented yet.
- flashlight: Profiles for flashlight.
- multiflashlight: Profiles for multiflashlight.

References

[1] Friedman J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29:1189–1232. [2] Apley D. W. (2016). Visualizing the effects of predictor variables in black box supervised learning models. ArXiv <arXiv:1612.08468>.

See Also

light_effects, plot.light_profile.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))

light_profile(mod_full, v = "Species")
light_profile(mod_full, v = "Species", counts = FALSE)
light_profile(mod_full, v = "Species", type = "response")
light_profile(mod_full, v = "Species", type = "ale")
light_profile(mod_full, v = "Species", stats = "quartiles")

light_profile(mod_full, v = "Petal.Width")
light_profile(mod_full, v = "Petal.Width", type = "residual")
light_profile(mod_full, v = "Petal.Width", type = "residual", v_label = FALSE)
light_profile(mod_full, v = "Petal.Width", type = "residual", dig.lab = 1)
light_profile(mod_full, v = "Petal.Width", stats = "quartiles")
light_profile(mod_full, v = "Petal.Width", n_bins = 3)
light_profile(mod_full, v = "Petal.Width", pd_evaluate_at = 2:4)
light_profile(mod_full, pd_grid = data.frame(Petal.Width = 2:4))

light_profile(mod_full, v = "Petal.Width", by = "Species")

light_profile(mods, v = "Petal.Width")
```

```
light_profile(mods, v = "Petal.Width", by = "Species")
light_profile(mods, v = "Petal.Width", by = "Species", type = "predicted")
light_profile(mods, v = "Petal.Width", by = "Species",
  type = "predicted", stats = "quartiles")

light_profile(mods, v = "Petal.Width", by = "Species", stats = "quartiles",
  value_name = "pd", q1_name = "p25", q3_name = "p75", label_name = "model",
  type_name = "visualization", counts_name = "n")
```

`light_recode` *Recode Factor Columns*

Description

Recodes factor levels of columns in data slots of an object of class `light`.

Usage

```
light_recode(x, ...)

## Default S3 method:
light_recode(x, ...)

## S3 method for class 'light'
light_recode(x, what, levels, labels, ...)
```

Arguments

`x`	An object of class `light`.
`...`	Further arguments passed to `factor`.
`what`	Column identifier in `x` (not column name) to be recoded, e.g. "type_name", "label_name".
`levels`	Current levels/values of `type_name` column (in desired order).
`labels`	New levels of `type_name` column in same order as `levels`.

Value

`x` with new factor levels of `type_name` column.

Methods (by class)

- `default`: Default method not implemented yet.
- `light`: Recoding factors in data slots of `light` object.

See Also

`plot.light_effects`.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))
eff <- light_effects(mods, v = "Species")
eff <- light_recode(eff, what = "type_name",
  levels = c("response", "predicted", "partial dependence"),
  labels = c("Observed", "Fitted", "Effect"))
plot(eff)
perf <- light_performance(mods)
perf <- light_recode(perf, what = "label_name",
  levels = c("part", "full"), labels = c("simple", "complex"))
plot(perf)
```

---

midpoints *Midpoints*

Description

Takes a vector of breaks and calculates midpoints of subsequent unique breaks.

Usage

```
midpoints(breaks)
```

Arguments

breaks	Numeric vector of cut points or a single number specifying the number of intervals desired.

Value

Vector of the same length as x minus 1 with midpoints of breaks.

Examples

```
midpoints(1:4)
midpoints(c(4, 4:1))
```

`most_important` *Most Important Variables.*

Description

Returns the most important variable names sorted descendingly.

Usage

```
most_important(x, top_m = Inf)

## Default S3 method:
most_important(x, top_m = Inf)

## S3 method for class 'light_importance'
most_important(x, top_m = Inf)
```

Arguments

`x`	An object of class `light_importance`.
`top_m`	Maximum number of important variables to be returned. Defaults to `Inf`, i.e. return all variables in descending order of importance.

Value

A character vector of variable names sorted in descending order by importance.

Methods (by class)

- `default`: Default method not implemented yet.
- `light_importance`: Extracts most important variables from an object of class `light_importance`.

See Also

`light_importance`.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
fl <- flashlight(model = fit, label = "ols", data = iris, y = "Sepal.Length")
(imp <- light_importance(fl, seed = 4))
most_important(imp)
most_important(imp, 2)
```

`multiflashlight` *Create or Update a multiflashlight*

Description

Combines a list of flashlights to an object of class `multiflashlight` and/or updates a multiflashlight.

Usage

```
multiflashlight(x, ...)

## Default S3 method:
multiflashlight(x, ...)

## S3 method for class 'flashlight'
multiflashlight(x, ...)

## S3 method for class 'list'
multiflashlight(x, ...)

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

Arguments

`x`	An object of class `multiflashlight`, `flashlight` or a list of flashlights.
`...`	Optional arguments in the flashlights to update, see examples.

Value

An object of class `multiflashlight`. This is a named list of flashlight objects.

Methods (by class)

- `default`: Used to create a flashlight object. No `x` has to be passed in this case.
- `flashlight`: Updates an existing flashlight object and turns into a multiflashlight.
- `list`: Creates (and updates) a multiflashlight from a list of flashlights.
- `multiflashlight`: Updates an object of class `multiflashlight`.

See Also

`flashlight`.

Examples

```
fit_lm <- lm(Sepal.Length ~ ., data = iris)
fit_glm <- glm(Sepal.Length ~ ., family = Gamma(link = log), data = iris)
mod_lm <- flashlight(model = fit_lm, label = "lm")
mod_glm <- flashlight(model = fit_glm, label = "glm")
(mods <- multiflashlight(list(mod_lm, mod_glm)))

mods <- multiflashlight(list(mod_lm, mod_glm),
  data = iris, by = "Species", y = "Sepal.Length")
mod_lm <- mods$lm
mod_lm
```

---

plot.light_breakdown *Visualize Variable Contribution Breakdown for Single Observation*

Description

Minimal visualization of an object of class `light_breakdown` as waterfall plot. The object returned is of class `ggplot` and can be further customized.

Usage

```
## S3 method for class 'light_breakdown'
plot(x, facet_scales = "free",
  facet_ncol = 1, ...)
```

Arguments

`x`	An object of class `light_breakdown`.
`facet_scales`	Scales argument passed to `facet_wrap`.
`facet_ncol`	`ncol` argument passed to `facet_wrap`.
`...`	Further arguments passed to `geom_label`.

Details

The waterfall plot is to be read from top to bottom. The first line describes the (weighted) average prediction in the query data used to start with. Then, each additional line shows how the prediction changes due to the impact of the corresponding variable. The last line finally shows the original prediction of the selected observation. Multiple flashlights are shown in different facets. Positive and negative impacts are visualized with different colors.

Value

An object of class `ggplot2`.

See Also

`light_importance`.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))

plot(x <- light_breakdown(mod_full, new_obs = iris[1, ]))
plot(light_breakdown(mods, new_obs = iris[1, ]), size = 2.5)
plot(light_breakdown(mods, new_obs = iris[1, ]), facet_ncol = 2)
```

---

plot.light_effects *Visualize Multiple Types of Profiles Together*

Description

Visualizes response-, prediction-, partial dependence, and/or ALE profiles of a (multi-)flashlight with respect to a covariable v. Different flashlights or a single flashlight with one "by" variable are separated by a facet wrap.

Usage

```
## S3 method for class 'light_effects'
plot(x, use = c("response", "predicted", "pd"),
  zero_counts = TRUE, size_factor = 1, facet_scales = "free_x",
  facet_nrow = 1L, rotate_x = TRUE, ...)
```

Arguments

x	An object of class light_effects.
use	A vector of elements to show. Any subset of ("response", "predicted", "pd" and "ale") or "all". Defaults to all except "ale"
zero_counts	Logical flag if 0 count levels should be shown on the x axis.
size_factor	Factor used to enlarge default size in geom_point and geom_line.
facet_scales	Scales argument passed to facet_wrap.
facet_nrow	Number of rows in facet_wrap. Must be 1 if plot_counts should be used.
rotate_x	Should x axis labels be rotated by 45 degrees?
...	Further arguments passed to geoms.

Value

An object of class ggplot2.

See Also

light_effects, plot_counts.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- glm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris,
  y = "Sepal.Length", w = "Petal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris,
  y = "Sepal.Length", w = "Petal.Length")
mods <- multiflashlight(list(mod_full, mod_part))

plot(light_effects(mod_full, v = "Species"))
x <- light_effects(mod_full, v = "Petal.Width")
plot(x)
plot(x, use = "response")
plot(x, use = c("pd", "ale"))
plot_counts(plot(x, use = c("pd", "ale")), x)

x <- light_effects(mod_full, v = "Petal.Width", stats = "quartiles")
plot(x)
plot_counts(plot(x, use = "response"), x, alpha = 0.2)

x <- light_effects(mod_full, v = "Petal.Width", by = "Species")
plot(x)
p <- plot(x, zero_counts = FALSE, use = "all")
plot_counts(p, x, alpha = 0.2)

plot(light_effects(mod_full, v = "Petal.Width", by = "Species",
  stats = "quartiles"))
plot(light_effects(mods, v = "Petal.Width", stats = "quartiles"))
```

---

plot.light_ice *Visualize ICE profiles*

Description

Minimal visualization of an object of class `light_ice` as `geom_line`. The object returned is of class `ggplot` and can be further customized.

Usage

```
## S3 method for class 'light_ice'
plot(x, facet_scales = "fixed", rotate_x = FALSE,
  ...)
```

Arguments

`x`	An object of class `light_ice`.
`facet_scales`	Scales argument passed to `facet_wrap`.
`rotate_x`	Should x axis labels be rotated by 45 degrees? Default is FALSE.
`...`	Further arguments passed to `geom_line`.

Details

Each observation is visualized by a line. The first "by" variable is represented by the color, a second "by" variable or a multiflashlight by facets.

Value

An object of class ggplot2.

Author(s)

Michael Mayer

See Also

light_ice.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))
grid <- expand.grid(Species = levels(iris$Species), Petal.Length = 2:4)

plot(light_ice(mod_full, v = "Species"), alpha = 0.2)
indices <- (1:15) * 10
plot(light_ice(mod_full, v = "Species"), rotate_x = TRUE)
plot(light_ice(mods, v = "Species", indices = indices))
plot(light_ice(mods, v = "Species", indices = indices, center = TRUE))
plot(light_ice(mods, v = "Petal.Width", n_bins = 5, indices = indices))
plot(light_ice(mods, v = "Petal.Width", by = "Species", n_bins = 5, indices = indices))

ir <- iris
ir$log_sl <- log(ir$Sepal.Length)
fit_lm <- lm(log_sl ~ Petal.Length + Petal.Width, data = ir)
fit_glm <- glm(Sepal.Length ~ Petal.Length + Petal.Width,
  data = ir, family = Gamma(link = log))
fl_lm <- flashlight(model = fit_lm, label = "lm", y = "log_sl", linkinv = exp)
fl_glm <- flashlight(model = fit_glm, label = "glm", y = "Sepal.Length",
  predict_function = function(m, X) predict(m, X, type = "response"))
fls <- multiflashlight(list(fl_lm, fl_glm), data = ir)
plot(light_ice(fls, v = "Petal.Length", indices = indices))
plot(light_ice(fls, v = "Petal.Length", indices = indices, center = TRUE))
plot(light_ice(fls, v = "Petal.Length", indices = indices, by = "Species", center = TRUE))
plot(light_ice(fls, v = "Petal.Length", indices = indices, use_linkinv = FALSE))
```

plot.light_importance *Visualize Model Importance*

Description

Minimal visualization of an object of class light_importance as geom_bar. The object returned is of class ggplot and can be further customized.

Usage

```
## S3 method for class 'light_importance'
plot(x, top_m = Inf, swap_dim = FALSE,
  facet_scales = "fixed", ...)
```

Arguments

x	An object of class light_importance.
top_m	Maximum number of important variables to be returned.
swap_dim	If multiflashlight and one "by" variable or single flashlight with two "by" variables, swap the role of dodge/fill variable and facet variable. If multiflashlight or one "by" variable, use facets instead of colors.
facet_scales	Scales argument passed to facet_wrap.
...	Further arguments passed to geom_bar.

Details

The plot is organized as a bar plot with variable names as x-aesthetic. Up to two additional dimensions (multiflashlight and one "by" variable or single flashlight with two "by" variables) can be visualized by facetting and dodge/fill. Set swap_dim = FALSE to revert the role of these two dimensions. One single additional dimension is visualized by a facet wrap, or - if swap_dim = FALSE - by dodge/fill.

Value

An object of class ggplot2.

See Also

light_importance.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part), by = "Species")
```

```
plot(light_importance(mod_full), fill = "darkred")
plot(light_importance(mod_full, variable_name = "v", label_name = "model",
  metric_name = "m", value_name = "drop"))
plot(light_importance(mod_full), top_m = 2)
plot(light_importance(mods))
plot(light_importance(mods), swap_dim = TRUE)
plot(light_importance(mods, by = NULL), fill = "darkgreen")
plot(light_importance(mods, by = NULL), swap_dim = TRUE)
```

`plot.light_performance` *Visualize Model Performance*

Description

Minimal visualization of an object of class `light_performance` as `geom_bar`. The object returned has class `ggplot` and can be further customized.

Usage

```
## S3 method for class 'light_performance'
plot(x, swap_dim = FALSE,
  facet_scales = "free_y", rotate_x = FALSE, ...)
```

Arguments

`x`	An object of class `light_performance`.
`swap_dim`	Should representation of dimensions (either two "by" variables or one "by" variable and multiflashlight) of x aesthetic and dodge fill aesthetic be swapped? Default is FALSE.
`facet_scales`	Scales argument passed to `facet_wrap`.
`rotate_x`	Should x axis labels be rotated by 45 degrees? Default is FALSE.
`...`	Further arguments passed to `geom_bar`.

Details

The plot is organized as a bar plot as follows: For flashlights without "by" variable specified, a single bar is drawn. Otherwise, the "by" variable (or the flashlight label if there is no "by" variable) is represented by the "x" aesthetic. The flashlight label (in case of one "by" variable) is represented by dodged bars. This strategy makes sure that performance of different flashlights can be compared easiest. Set "swap_dim = TRUE" to revert the role of dodging and x aesthetic. Different metrics are always represented by facets.

Value

An object of class `ggplot2`.

See Also

light_performance.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))
plot(light_performance(mod_full), fill = "darkred")
plot(light_performance(mod_full, by = "Species"), fill = "darkred")
plot(light_performance(mods))
plot(light_performance(mods, by = "Species"))
plot(light_performance(mods, by = "Species"), swap_dim = TRUE)
```

---

plot.light_profile *Visualize Profiles, e.g. of Partial Dependence*

Description

Minimal visualization of an object of class light_profile. The object returned is of class ggplot and can be further customized.

Usage

```
## S3 method for class 'light_profile'
plot(x, swap_dim = FALSE,
  facet_scales = "free_x", rotate_x = x$type != "partial dependence",
  ...)
```

Arguments

x	An object of class light_profile.
swap_dim	If multiflashlight and one "by" variable or single flashlight with two "by" variables, swap the role of dodge/fill variable and facet variable. If multiflashlight or one "by" variable, use facets instead of colors.
facet_scales	Scales argument passed to facet_wrap.
rotate_x	Should x axis labels be rotated by 45 degrees? TRUE, except for type "partial dependence".
...	Further arguments passed to geom_point and geom_line.

Details

Either lines and points are plotted (if stats = "mean") or quartile boxes. If there is a "by" variable or a multiflashlight, this first dimension is taken care by color (or if swap_dim = TRUE by facets). If there are two "by" variables or a multiflashlight with one "by" variable, the first "by" variable is visualized as color, the second one or the multiflashlight via facet (change with swap_dim).

Value

An object of class ggplot2.

See Also

light_profile, plot.light_effects.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- lm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))

plot(light_profile(mod_full, v = "Species"))
plot(light_profile(mod_full, v = "Species", type = "ale"))
plot(light_profile(mod_full, v = "Species", type = "response"))
plot(light_profile(mod_full, v = "Species", type = "residual", stats = "quartiles"))

plot(light_profile(mod_full, v = "Petal.Width"))
plot(light_profile(mod_full, v = "Petal.Width", type = "residual"))
plot(light_profile(mod_full, v = "Petal.Width", stats = "quartiles"))

plot(light_profile(mod_full, v = "Petal.Width", by = "Species"))
plot(light_profile(mod_full, v = "Petal.Width", by = "Species"), swap_dim = TRUE)

plot(light_profile(mods, v = "Species"))
plot(light_profile(mods, v = "Petal.Width"))
plot(light_profile(mods, v = "Petal.Width"), swap_dim = TRUE)
plot(light_profile(mods, v = "Petal.Width", by = "Species"))
plot(light_profile(mods, v = "Petal.Width", by = "Species", type = "ale"))
plot(light_profile(mods, v = "Petal.Width", by = "Species"), swap_dim = TRUE)
plot(light_profile(mods, v = "Petal.Width", by = "Species", type = "predicted"))
plot(light_profile(mods, v = "Petal.Width", by = "Species",
  type = "predicted", n_bins = 5), swap_dim = TRUE)
plot(light_profile(mods, v = "Petal.Width", by = "Species",
  type = "predicted", stats = "quartiles"), rotate_x = TRUE)
```

---

plot_counts *Add Counts to Effects Plot*

Description

Add counts as bar plot on top of light_effects plot.

Usage

```
plot_counts(p, x, text_size = 3, facet_scales = "free_x",
  show_labels = TRUE, big.mark = "'", scientific = FALSE, ...)
```

Arguments

p	The result of plot.light_effects.
x	An object of class light_effects.
text_size	Size of count labels.
facet_scales	Scales argument passed to facet_wrap.
show_labels	Should counts be added as text?
big.mark	Parameter passed to format. Default is "'".
scientific	Parameter passed to format. Default is FALSE.
...	Further arguments passed to geom_bar.

Details

Experimental. Uses package ggpubr to rearrange the figure. Thus, the resulting plot cannot be easily modified. Furthermore, adding counts only works if the legend in plot.light_effects is not placed on the left or right side of the plot. It has to be placed inside or at the bottom.

Value

An object of class ggplot2.

See Also

plot.light_effects.

Examples

```
fit_full <- lm(Sepal.Length ~ ., data = iris)
fit_part <- glm(Sepal.Length ~ Petal.Length, data = iris)
mod_full <- flashlight(model = fit_full, label = "full", data = iris, y = "Sepal.Length")
mod_part <- flashlight(model = fit_part, label = "part", data = iris, y = "Sepal.Length")
mods <- multiflashlight(list(mod_full, mod_part))

x <- light_effects(mod_full, v = "Petal.Width", stats = "quartiles")
plot_counts(plot(x), x, width = 0.3, alpha = 0.2)
plot_counts(plot(x, zero_counts = FALSE), x, width = 0.3, alpha = 0.2)
plot_counts(plot(x), x, width = 0.3, alpha = 0.2, show_labels = FALSE)
plot_counts(plot(x, use = "response"), x, fill = "lightblue")
plot_counts(plot(x, use = "pd", show.legend = FALSE), x, fill = "lightblue")
```

predict.flashlight *Predictions for flashlight*

Description

Predict method for an object of class `flashlight`. Pass additional elements to update the flashlight, typically `data`.

Usage

```
## S3 method for class 'flashlight'
predict(object, ...)
```

Arguments

`object`	An object of class `flashlight`.
`...`	Arguments used to update the flashlight.

Value

A vector with predictions.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
(fl <- flashlight(model = fit, data = iris, y = "Sepal.Length", label = "ols"))
predict(fl)[1:5]
predict(fl, data = iris[1:5, ])
predict(fl, data = iris[1:5, ], linkinv = exp)
```

print.flashlight *Prints a flashlight*

Description

Print method for an object of class `flashlight`.

Usage

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

Arguments

`x`	A on object of class `flashlight`.
`...`	Further arguments passed from other methods.

Value

Invisibly, the input is returned.

See Also

flashlight.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
flashlight(model = fit, label = "lm", y = "Sepal.Length", data = iris)
```

---

print.light *Prints light Object*

Description

Print method for an object of class `light`.

Usage

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

Arguments

x	A on object of class `light`.
...	Further arguments passed from other methods.

Value

Invisibly, the input is returned.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
fl <- flashlight(model = fit, label = "lm", y = "Sepal.Length", data = iris)
light_performance(fl, v = "Species")
light_effects(fl, v = "Sepal.Length")
```

print.multiflashlight *Prints a multiflashlight*

Description

Print method for an object of class `multiflashlight`.

Usage

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

Arguments

`x`	An object of class `multiflashlight`.
`...`	Further arguments passed to `print.flashlight`.

Value

Invisibly, the input is returned.

See Also

`multiflashlight`.

Examples

```
fit_lm <- lm(Sepal.Length ~ ., data = iris)
fit_glm <- glm(Sepal.Length ~ ., family = Gamma(link = log), data = iris)
fl_lm <- flashlight(model = fit_lm, label = "lm")
fl_glm <- flashlight(model = fit_glm, label = "glm")
multiflashlight(list(fl_lm, fl_glm), data = iris)
```

residuals.flashlight *Residuals for flashlight*

Description

Residuals method for an object of class `flashlight`. Pass additional elements to update the flashlight before calculation of residuals.

Usage

```
## S3 method for class 'flashlight'
residuals(object, ...)
```

Arguments

object	An object of class flashlight.
...	Arguments used to update the flashlight before calculating the residuals.

Value

A numeric vector with residuals.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
(fl <- flashlight(model = fit, data = iris, y = "Sepal.Length", label = "ols"))
residuals(fl)[1:5]
residuals(fl, data = iris[1:5, ])
residuals(fl, data = iris[1:5, ], linkinv = exp)
resid(fl)[1:5]
```

---

response *Response of flashlight*

Description

Extracts response from object of class flashlight.

Usage

```
response(object, ...)

## Default S3 method:
response(object, ...)

## S3 method for class 'flashlight'
response(object, ...)
```

Arguments

object	An object of class flashlight.
...	Arguments used to update the flashlight before extracting the response.

Value

A numeric vector of responses.

Methods (by class)

- default: Default method not implemented yet.
- flashlight: Extract response from flashlight object.

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
(fl <- flashlight(model = fit, data = iris, y = "Sepal.Length", label = "ols"))
response(fl)[1:5]
response(fl, data = iris[1:5, ])
response(fl, data = iris[1:5, ], linkinv = exp)
```

Index