Package 'flashlight'

October 13, 2025

Title Shed Light on Black Box Machine Learning Models

Version 1.0.0

Description Shed light on black box machine learning models by the help of model performance, variable importance, global surrogate models, ICE profiles, partial dependence (Friedman J. H. (2001) <doi:10.1214/aos/1013203451>), accumulated local effects (Apley D. W. (2016) <doi:10.48550/arXiv.1612.08468>), further effects plots, interaction strength, and variable contribution breakdown (Gosiewska and Biecek (2019) <doi:10.48550/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 (>= 3.2.0)

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr (>= 1.1.0), ggplot2, MetricsWeighted (>= 0.3.0), rlang (>= 0.3.0), rpart, rpart.plot, stats, tibble, tidyr (>= 1.0.0), tidyselect, utils

URL https://github.com/mayer79/flashlight

BugReports https://github.com/mayer79/flashlight/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Michael Mayer [aut, cre, cph]

Maintainer Michael Mayer <mayermichael79@gmail.com>

Repository CRAN

Date/Publication 2025-10-13 15:40:02 UTC

Contents

add_shap *DEPRECATED*

Description

Deprecated in favor of kernelshap/fastshap/shapr.

Usage

```
add_shap(...)
```

Arguments

...	Deprecated

Value

Error message.

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 = stats::predict,
  linkinv = function(z) z,
  w = NULL,
  by = NULL,
  metrics = list(rmse = MetricsWeighted::rmse),
  label = NULL,
  shap = NULL,
  ...
```

```
)

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

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.
shap	An optional shap object. Typically added by calling add_shap().
check	When updating the flashlight: Should internal checks be performed? Default is TRUE.

Value

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

Methods (by class)

- flashlight(default): Used to create a flashlight object. No x has to be passed in this case.
- flashlight(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))
```

`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_profile2d(x)

is.light_profile2d_multi(x)

is.light_effects(x)

is.light_effects_multi(x)

is.shap(x)

is.light_scatter(x)
```

```
is.light_scatter_multi(x)

is.light_global_surrogate(x)

is.light_global_surrogate_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_profile2d()`: Check for light_profile2d object.
- `is.light_profile2d_multi()`: Check for light_profile2d_multi object.
- `is.light_effects()`: Check for light_effects object.
- `is.light_effects_multi()`: Check for light_effects_multi object.
- `is.shap()`: Check for shap object.
- `is.light_scatter()`: Check for light_scatter object.
- `is.light_scatter_multi()`: Check for light_scatter_multi object.
- `is.light_global_surrogate()`: Check for light_global_surrogate object.
- `is.light_global_surrogate_multi()`: Check for light_global_surrogate_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 contributions (approximate SHAP) to the prediction of a single observation, see Gosiewska and Biecek (see reference) 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,
  visit_strategy = c("importance", "permutation", "v"),
  n_max = Inf,
  n_perm = 20,
  seed = NULL,
  use_linkinv = FALSE,
  description = TRUE,
  digits = 2,
  ...
)

## 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 variable names to assess contribution for. Defaults to all except those specified by "y", "w" and "by".

visit_strategy	In what sequence should variables be visited? By "importance", by n_perm "permutation" or as "v" (see Details).
n_max	Maximum number of rows in data to consider in the reference data. Set to lower value if data is large.
n_perm	Number of permutations of random visit sequences. Only used if visit_strategy = "permutation".
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.
description	Should descriptions be added? Default is TRUE.
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 the reference below for an explanation. Unfortunately, there is no efficient way to do this in a model agnostic way.

We offer two visit strategies to approximate SHAP:

1. "importance": Using the short-cut described in the reference below: 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.
2. "permutation": Averages contributions from a small number of random permutations of v.

Note that the minimum required elements in the (multi-)flashlight are a "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" with the following elements:

- data A tibble with results.
- by Same as input by.

Methods (by class)

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

References

A. Gosiewska and P. Biecek (2019). IBREAKDOWN: Uncertainty of model explanations for non-additive predictive models. ArXiv.

See Also

plot.light_breakdown()

Examples

```
fit_part <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
fl_part <- flashlight(
  model = fit_part, label = "part", data = iris, y = "Sepal.Length"
)
plot(light_breakdown(fl_part, new_obs = iris[1, ]))

# Second model
fit_full <- lm(Sepal.Length ~ ., data = iris)
fl_full <- flashlight(
  model = fit_full, label = "full", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_part, fl_full))
plot(light_breakdown(fls, new_obs = iris[1, ]))
```

---

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".
...	Further arguments passed from or to other methods.

Value

The input x or an error message.

Methods (by class)

- light_check(default): Default check method not implemented yet.
- light_check(flashlight): Checks if a flashlight object is consistently defined.
- light_check(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)

- light_combine(default): Default method not implemented yet.
- light_combine(light): Since there is nothing to combine, the input is returned except for additional classes.
- light_combine(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_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 = "mean",
  breaks = NULL,
  n_bins = 11L,
  cut_type = c("equal", "quantile"),
  use_linkinv = TRUE,
  counts_weighted = FALSE,
  v_labels = TRUE,
  pred = NULL,
  pd_indices = NULL,
  pd_n_max = 1000L,
  pd_seed = NULL,
  ale_two_sided = TRUE,
  ...
)

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

Arguments

x	An object of class "flashlight" or "multiflashlight".
...	Further arguments passed to formatC() in forming the cut breaks of the v variable.
v	The variable name to be profiled.
data	An optional data.frame.
by	An optional vector of column names used to additionally group the results.
stats	Deprecated. Will be removed in version 1.1.0.
breaks	Cut breaks for a numeric v. Used to overwrite automatic binning via n_bins and cut_type. Ignored if v is not numeric.
n_bins	Approximate number of unique values to evaluate for numeric v. Ignored if v is not numeric or if breaks is specified.

cut_type	Should a numeric v be cut into "equal" or "quantile" bins? Ignored if v is not numeric or if breaks is specified.
use_linkinv	Should retransformation function be applied? Default is TRUE.
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 for instance when 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. Not implemented for multiflashlight.
pd_indices	A vector of row numbers to consider in calculating partial dependence profiles and "ale".
pd_n_max	Maximum number of ICE profiles to calculate (will be randomly picked from data) for partial dependence and ALE.
pd_seed	Integer random seed used to select 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. More specifically: Usually, local effects at value x are calculated using points in $[x - e, x]$. Set ale_two_sided = TRUE to use points in $[x - e/2, x + e/2]$.

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 class "light_effects" with the following elements:

- response: A tibble containing the response profiles. Column names can be controlled by options(flashlight.column_name).
- predicted: A tibble containing the prediction profiles.
- pd: A tibble containing the partial dependence profiles.
- ale: A tibble containing the ALE profiles.
- by: Same as input by.
- v: The variable(s) evaluated.

Methods (by class)

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

See Also

light_profile(), plot.light_effects()

Examples

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

# PDP, average response, average predicted by Species
eff <- light_effects(fl_lin, v = "Petal.Length")
plot(eff)

# PDP and ALE
plot(eff, use = c("pd", "ale"), recode_labels = c(ale = "ALE"))

# Second model with non-linear Petal.Length effect
fit_nonlin <- lm(Sepal.Length ~ . + I(Petal.Length^2), data = iris)
fl_nonlin <- flashlight(
  model = fit_nonlin, label = "nonlin", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_lin, fl_nonlin))

# PDP and ALE
plot(light_effects(fls, v = "Petal.Length"), use = c("pd", "ale"))
```

---

light_global_surrogate

Global Surrogate Tree

Description

Model predictions are modelled by a single decision tree, serving as an easy to interprete surrogate to the original model. As suggested in Molnar (see reference below), the quality of the surrogate tree can be measured by its R-squared. The size of the tree can be modified by passing ... arguments to rpart::rpart().

Usage

```
light_global_surrogate(x, ...)

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

## S3 method for class 'flashlight'
light_global_surrogate(
  x,
  data = x$data,
  by = x$by,
```

```
  v = NULL,
  use_linkinv = TRUE,
  n_max = Inf,
  seed = NULL,
  keep_max_levels = 4L,
  ...
)

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

Arguments

x	An object of class "flashlight" or "multiflashlight".
...	Arguments passed to rpart::rpart(), such as maxdepth.
data	An optional data.frame.
by	An optional vector of column names used to additionally group the results. For each group, a separate tree is grown.
v	Vector of variables used in the surrogate model. Defaults to all variables in data except "by", "w" and "y".
use_linkinv	Should retransformation function be applied? Default is TRUE.
n_max	Maximum number of data rows to consider to build the tree.
seed	An integer random seed used to select data rows if n_max is lower than the number of data rows.
keep_max_levels	Number of levels of categorical and factor variables to keep. Other levels are combined to a level "Other". This prevents rpart::rpart() to take too long to split non-numeric variables with many levels.

Value

An object of class "light_global_surrogate" with the following elements:

- data A tibble with results.
- by Same as input by.

Methods (by class)

- light_global_surrogate(default): Default method not implemented yet.
- light_global_surrogate(flashlight): Surrogate model for a flashlight.
- light_global_surrogate(multiflashlight): Surrogate model for a multiflashlight.

References

Molnar C. (2019). Interpretable Machine Learning.

See Also

plot.light_global_surrogate()

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
x <- flashlight(model = fit, label = "lm", data = iris)
sur <- light_global_surrogate(x)
sur$data$r_squared
plot(sur)
```

---

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 (see the reference below for details). The curves can be centered in order to increase visibility of interaction effects.

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 = 27L,
  cut_type = c("equal", "quantile"),
  indices = NULL,
  n_max = 20L,
  seed = NULL,
  use_linkinv = TRUE,
  center = c("no", "first", "middle", "last", "mean", "0"),
  ...
)
```

```
## 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 name 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	Cut breaks for a numeric v. Used to overwrite automatic binning via n_bins and cut_type. Ignored if v is not numeric or if grid or evaluate_at are specified.
grid	A data.frame with evaluation grid. For instance, can be generated by expand.grid().
n_bins	Approximate number of unique values to evaluate for numeric v. Ignored if v is not numeric or if breaks, grid or evaluate_at are specified.
cut_type	Should a numeric v be cut into "equal" or "quantile" bins? Ignored if v is not numeric or if breaks, grid or evaluate_at are 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	How should curves be centered? • Default is "no". • Choose "first", "middle", or "last" to 0-center at specific evaluation points. • Choose "mean" to center all profiles at the within-group means. • Choose "0" to mean-center curves at 0.

Details

There are two ways to specify the variable(s) to be profiled.

1. 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.
2. More general: Specify any grid as a data.frame with one or more columns. For instance, it can be generated by a call to expand.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 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" with the following elements:

- `data` A tibble containing the results.
- by Same as input by.
- v The variable(s) evaluated.
- `center` How centering was done.

Methods (by class)

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

References

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_add <- lm(Sepal.Length ~ ., data = iris)
fl_add <- flashlight(model = fit_add, label = "additive", data = iris)

plot(light_ice(fl_add, v = "Sepal.Width", n_max = 200), alpha = 0.2)
plot(light_ice(fl_add, v = "Sepal.Width", n_max = 200, center = "first"))

# Second model with interactions
fit_nonadd <- lm(Sepal.Length ~ . + Sepal.Width:Species, data = iris)
fl_nonadd <- flashlight(model = fit_nonadd, label = "nonadditive", data = iris)
fls <- multiflashlight(list(fl_add, fl_nonadd))

plot(light_ice(fls, v = "Sepal.Width", by = "Species", n_max = 200), alpha = 0.2)
plot(light_ice(fls, v = "Sepal.Width", by = "Species", n_max = 200, center = "mid"))
```

light_importance *Permutation Variable Importance*

Description

Importance of variable v is measured as drop in performance by permuting the values of v, see Fisher et al. 2018 (reference below).

Usage

```
light_importance(x, ...)

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

## S3 method for class 'flashlight'
light_importance(
  x,
  data = x$data,
  by = x$by,
  type = c("permutation", "shap"),
  v = NULL,
  n_max = Inf,
  seed = NULL,
  m_repetitions = 1L,
  metric = x$metrics[1L],
  lower_is_better = TRUE,
  use_linkinv = FALSE,
  ...
)

## 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.
type	Type of importance: "permutation" (currently the only option).
v	Vector of variable names to assess importance for. Defaults to all variables in data except "by" and "y".
n_max	Maximum number of rows to consider.
seed	An integer random seed used to select and shuffle rows.

m_repetitions	Number of permutations. Defaults to 1. A value above 1 provides more stable estimates of variable importance and allows the calculation of standard errors measuring the uncertainty from permuting.
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 ....
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.

Details

The minimum required elements in the (multi-)flashlight are "y", "predict_function", "model", "data" and "metrics".

Value

An object of class "light_importance" with the following elements:

- data A tibble with results.
- by Same as input by.
- type Same as input type. For information only.

Methods (by class)

- light_importance(default): Default method not implemented yet.
- light_importance(flashlight): Variable importance for a flashlight.
- light_importance(multiflashlight): Variable importance for a multiflashlight.

References

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.

See Also

most_important(), plot.light_importance()

Examples

```
fit_part <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
fl_part <- flashlight(
  model = fit_part, label = "part", data = iris, y = "Sepal.Length"
)

# No effect of some variables (incl. standard errors)
plot(light_importance(fl_part, m_repetitions = 4), fill = "chartreuse4")
```

```
# Second model includes all variables
fit_full <- lm(Sepal.Length ~ ., data = iris)
fl_full <- flashlight(
  model = fit_full, label = "full", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_part, fl_full))

plot(light_importance(fls), fill = "chartreuse4")
plot(light_importance(fls, by = "Species"))
```

`light_interaction` *Interaction Strength*

Description

This function provides Friedman's H statistic for overall interaction strength per covariable as well as its version for pairwise interactions, see the reference below.

Usage

```
light_interaction(x, ...)

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

## S3 method for class 'flashlight'
light_interaction(
  x,
  data = x$data,
  by = x$by,
  v = NULL,
  pairwise = FALSE,
  type = c("H", "ice"),
  normalize = TRUE,
  take_sqrt = TRUE,
  grid_size = 200L,
  n_max = 1000L,
  seed = NULL,
  use_linkinv = FALSE,
  ...
)

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

Arguments

x	An object of class "flashlight" or "multiflashlight".
...	Further arguments passed to or from other methods.
data	An optional data.frame.
by	An optional vector of column names used to additionally group the results.
v	Vector of variable names to be assessed.
pairwise	Should overall interaction strength per variable be shown or pairwise interactions? Defaults to FALSE.
type	Are measures based on Friedman's H statistic ("H") or on "ice" curves? Option "ice" is available only if pairwise = FALSE.
normalize	Should the variances explained be normalized? Default is TRUE in order to reproduce Friedman's H statistic.
take_sqrt	In order to reproduce Friedman's H statistic, resulting values are root transformed. Set to FALSE if squared values should be returned.
grid_size	Grid size used to form the outer product. Will be randomly picked from data (after limiting to n_max).
n_max	Maximum number of data rows to consider. Will be randomly picked from data if necessary.
seed	An integer random seed used for subsampling.
use_linkinv	Should retransformation function be applied? Default is FALSE.

Details

As a fast alternative to assess overall interaction strength, with type = "ice", the function offers a method based on centered ICE curves: The corresponding H* statistic measures how much of the variability of a c-ICE curve is unexplained by the main effect. As for Friedman's H statistic, it can be useful to consider unnormalized or squared values (see Details below).

Friedman's H statistic relates the interaction strength of a variable (pair) to the total effect strength of that variable (pair) based on partial dependence curves. Due to this normalization step, even variables with low importance can have high values for H. The function light_interaction() offers the option to skip normalization in order to have a more direct comparison of the interaction effects across variable (pairs). The values of such unnormalized H statistics are on the scale of the response variable. Use take_sqrt = FALSE to return squared values of H. Note that in general, for each variable (pair), predictions are done on a data set with grid_size * n_max, so be cautious with increasing the defaults too much. Still, even with larger grid_size and n_max, there might be considerable variation across different runs, thus, setting a seed is recommended.

The minimum required elements in the (multi-) flashlight are a "predict_function", "model", and "data".

Value

An object of class "light_importance" with the following elements:

- data A tibble containing the results. Can be used to build fully customized visualizations. Column names can be controlled by options(flashlight.column_name).

- by Same as input by.
- type Same as input type. For information only.

Methods (by class)

- light_interaction(default): Default method not implemented yet.
- light_interaction(flashlight): Interaction strengths for a flashlight object.
- light_interaction(multiflashlight): for a multiflashlight object.

References

Friedman, J. H. and Popescu, B. E. (2008). "Predictive learning via rule ensembles." The Annals of Applied Statistics. JSTOR, 916–54.

See Also

light_ice()

Examples

```
# First model with interactions
fit_nonadd <- lm(
  Sepal.Length ~ . + Sepal.Width:Species + Petal.Width:Species, data = iris
)
fl_nonadd <- flashlight(
  model = fit_nonadd, label = "nonadditive", data = iris, y = "Sepal.Length"
)

# Friedman's H per feature
plot(light_interaction(fl_nonadd), fill = "chartreuse4")

# Unnormalized H^2 measures proportion of bivariate effect explained by interaction
plot(
  light_interaction(fl_nonadd, normalize = TRUE, take_sqrt = TRUE),
  fill = "chartreuse4"
)

# Pairwise H
plot(light_interaction(fl_nonadd, pairwise = TRUE), fill = "chartreuse4")

# Second model without interactions
fit_add <- lm(Sepal.Length ~ ., data = iris)
fl_add <- flashlight(
  model = fit_add, label = "additive", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_add, fl_nonadd))

plot(light_interaction(fls), fill = "chartreuse4")
```

`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,
  ...
)

## 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`.

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" with the following elements:

- `data`: A tibble containing the results.
- by Same as input by.

Methods (by class)

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

See Also

`plot.light_performance()`

Examples

```
fit_part <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
fl_part <- flashlight(
  model = fit_part, label = "part", data = iris, y = "Sepal.Length"
)
plot(light_performance(fl_part, by = "Species"), fill = "chartreuse4")

# Second model
fit_full <- lm(Sepal.Length ~ ., data = iris)
fl_full <- flashlight(
  model = fit_full, label = "full", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_part, fl_full))

plot(light_performance(fls, by = "Species"))
plot(light_performance(fls, by = "Species"), swap_dim = TRUE)
```

---

light_profile *Partial Dependence and other Profiles*

Description

Calculates different types of profiles across covariable values. By default, partial dependence profiles are calculated (see Friedman). Other options are profiles of ALE (accumulated local effects, see Apley), response, predicted values ("M plots" or "marginal plots", see Apley), 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 Apley, 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", "shap"),
  stats = "mean",
  breaks = NULL,
  n_bins = 11L,
  cut_type = c("equal", "quantile"),
  use_linkinv = TRUE,
  counts = TRUE,
  counts_weighted = FALSE,
  v_labels = TRUE,
  pred = NULL,
  pd_evaluate_at = NULL,
  pd_grid = NULL,
  pd_indices = NULL,
  pd_n_max = 1000L,
  pd_seed = NULL,
  pd_center = c("no", "first", "middle", "last", "mean", "0"),
  ale_two_sided = FALSE,
  ...
)

## S3 method for class 'multiflashlight'
light_profile(
  x,
  v = NULL,
  data = NULL,
 type = c("partial dependence", "ale", "predicted", "response", "residual", "shap"),
  breaks = NULL,
  n_bins = 11L,
  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 formatC() in forming the cut breaks of the v variable.
v	The variable name 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	Deprecated. Will be removed in version 1.1.0.
breaks	Cut breaks for a numeric v. Used to overwrite automatic binning via n_bins and cut_type. Ignored if v is not numeric.
n_bins	Approximate number of unique values to evaluate for numeric v. Ignored if v is not numeric or if breaks is specified.
cut_type	Should a numeric v be cut into "equal" or "quantile" bins? Ignored if v is not numeric or if breaks is specified.
use_linkinv	Should retransformation function be applied? Default is TRUE.
counts	Should observation counts be added?
counts_weighted	If counts = 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, for instance, if 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. Not implemented for multiflashlight.
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 and "ale".
pd_n_max	Maximum number of ICE profiles to calculate (will be randomly picked from data) for partial dependence and ALE.
pd_seed	Integer random seed used to select ICE profiles for partial dependence and ALE.
pd_center	How should ICE curves be centered? • Default is "no". • Choose "first", "middle", or "last" to 0-center at specific evaluation points.

- Choose "mean" to center all profiles at the within-group means.
- Choose "0" to mean-center curves at 0. Only relevant for partial dependence.

`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 in $[x - e, x]$. Set `ale_two_sided = TRUE` to use points in $[x - e/2, x + e/2]$.

Details

Numeric covariables `v` with more than `n_bins` disjoint values are binned into `n_bins` bins. Alternatively, `breaks` can be provided to specify the binning. For partial dependence profiles (and partly also ALE profiles), this behaviour can be overwritten 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 "y", "linkinv" and "data". "data" can also be passed on the fly.

Value

An object of class "light_profile" with the following elements:

- `data` A tibble containing results.
- `by` Names of group by variable.
- `v` The variable(s) evaluated.
- `type` Same as input `type`. For information only.

Methods (by class)

- `light_profile(default)`: Default method not implemented yet.
- `light_profile(flashlight)`: Profiles for flashlight.
- `light_profile(multiflashlight)`: Profiles for multiflashlight.

References

- Friedman J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29:1189–1232.
- Apley D. W. (2016). Visualizing the effects of predictor variables in black box supervised learning models.

See Also

`light_effects()`, `plot.light_profile()`

Examples

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

# PDP by Species
plot(light_profile(fl_lin, v = "Petal.Length", by = "Species"))

# Average predicted
plot(light_profile(fl_lin, v = "Petal.Length", type = "pred"))

# Second model with non-linear Petal.Length effect
fit_nonlin <- lm(Sepal.Length ~ . + I(Petal.Length^2), data = iris)
fl_nonlin <- flashlight(
  model = fit_nonlin, label = "nonlin", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_lin, fl_nonlin))

# PDP by Species
plot(light_profile(fls, v = "Petal.Length", by = "Species"))
plot(light_profile(fls, v = "Petal.Length", by = "Species"), swap_dim = TRUE)

# Average residuals (calibration)
plot(light_profile(fls, v = "Petal.Length", type = "residual"))
```

---

light_profile2d *2D Partial Dependence and other 2D Profiles*

Description

Calculates different types of 2D-profiles across two variables. By default, partial dependence profiles are calculated (see Friedman). Other options are response, predicted values, and residuals. The results are aggregated by (weighted) means.

Usage

```
light_profile2d(x, ...)

## Default S3 method:
light_profile2d(x, ...)

## S3 method for class 'flashlight'
light_profile2d(
  x,
  v = NULL,
  data = NULL,
  by = x$by,
  type = c("partial dependence", "predicted", "response", "residual", "shap"),
  breaks = NULL,
```

```
  n_bins = 11L,
  cut_type = "equal",
  use_linkinv = TRUE,
  counts = TRUE,
  counts_weighted = FALSE,
  pd_evaluate_at = NULL,
  pd_grid = NULL,
  pd_indices = NULL,
  pd_n_max = 1000L,
  pd_seed = NULL,
  ...
)

## S3 method for class 'multiflashlight'
light_profile2d(
  x,
  v = NULL,
  data = NULL,
  type = c("partial dependence", "predicted", "response", "residual", "shap"),
  breaks = NULL,
  n_bins = 11L,
  cut_type = "equal",
  pd_evaluate_at = NULL,
  pd_grid = NULL,
  ...
)
```

Arguments

x	An object of class "flashlight" or "multiflashlight".
...	Further arguments passed to formatC() in forming the cut breaks of the v variables. Not relevant for partial dependence profiles.
v	A vector of exactly two variable names 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", "predicted", "response", or "residual".
breaks	Named list of cut breaks specifying how to bin one or more numeric variables. Used to overwrite automatic binning via n_bins and cut_type. Ignored for non-numeric v.
n_bins	Approximate number of unique values to evaluate for numeric v. Can be an unnamed vector of length 2 to distinguish between v.
cut_type	Should numeric v be cut into "equal" or "quantile" bins? Can be an unnamed vector of length 2 to distinguish between v.
use_linkinv	Should retransformation function be applied? Default is TRUE.
counts	Should observation 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.

pd_evaluate_at
: An named list of evaluation points for one or more variables. Only relevant for type = "partial dependence".

pd_grid
: An evaluation data.frame with exactly two columns, e.g., generated by expand.grid(). Only used for type = "partial dependence". Offers maximal flexibility.

pd_indices
: A vector of row numbers to consider in calculating partial dependence profiles. Only used for type = "partial dependence".

pd_n_max
: Maximum number of ICE profiles to calculate (will be randomly picked from data). Only used for type = "partial dependence".

pd_seed
: Integer random seed used to select ICE profiles. Only used for type = "partial dependence".

Details

Different binning options are available, see arguments below. For high resolution partial dependence plots, it might be necessary to specify breaks, pd_evaluate_at or pd_grid in order to avoid empty parts in the plot. A high value of n_bins might not have the desired effect as it internally capped at the number of distinct values of a variable.

For partial dependence and prediction profiles, "model", "predict_function", "linkinv" and "data" are required. For response profiles it is "y", "linkinv" and "data". "data" can also be passed on the fly.

Value

An object of class "light_profile2d" with the following elements:

- data A tibble containing results.
- by Names of group by variables.
- v The two variable names evaluated.
- type Same as input type. For information only.

Methods (by class)

- light_profile2d(default): Default method not implemented yet.
- light_profile2d(flashlight): 2D profiles for flashlight.
- light_profile2d(multiflashlight): 2D profiles for multiflashlight.

References

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

See Also

light_profile(), plot.light_profile2d()

Examples

```
fit_part <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
fl_part <- flashlight(
  model = fit_part, label = "part", data = iris, y = "Sepal.Length"
)

# No effect of Petal.Width
plot(light_profile2d(fl_part, v = c("Petal.Length", "Petal.Width")))

# Second model includes Petal.Width
fit_full <- lm(Sepal.Length ~ ., data = iris)
fl_full <- flashlight(
  model = fit_full, label = "full", data = iris, y = "Sepal.Length"
)
fls <- multiflashlight(list(fl_part, fl_full))

plot(light_profile2d(fls, v = c("Petal.Length", "Petal.Width")))
```

---

light_recode *DEPRECATED*

Description

DEPRECATED

Recode Factor Columns - DEPRECATED

Usage

```
light_recode(...)

light_recode(...)
```

Arguments

...	Deprecated.

Value

Error message.

Deprecated.

`light_scatter` *Scatter Plot Data*

Description

This function prepares values for drawing a scatter plot of predicted values, responses, or residuals against a selected variable.

Usage

```
light_scatter(x, ...)

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

## S3 method for class 'flashlight'
light_scatter(
  x,
  v,
  data = x$data,
  by = x$by,
  type = c("predicted", "response", "residual", "shap"),
  use_linkinv = TRUE,
  n_max = 400,
  seed = NULL,
  ...
)

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

Arguments

`x`	An object of class "flashlight" or "multiflashlight".
`...`	Further arguments passed from or to other methods.
`v`	The variable name to be shown on the x-axis.
`data`	An optional `data.frame`.
`by`	An optional vector of column names used to additionally group the results.
`type`	Type of the profile: Either "predicted", "response", or "residual".
`use_linkinv`	Should retransformation function be applied? Default is `TRUE`.
`n_max`	Maximum number of data rows to select. Will be randomly picked.
`seed`	An integer random seed used for subsampling.

Value

An object of class "light_scatter" with the following elements:

- `data`: A tibble with results.
- `by`: Same as input `by`.
- `v`: The variable name evaluated.
- `type`: Same as input `type`. For information only.

Methods (by class)

- `light_scatter(default)`: Default method not implemented yet.
- `light_scatter(flashlight)`: Variable profile for a flashlight.
- `light_scatter(multiflashlight)`: light_scatter for a multiflashlight.

See Also

`plot.light_scatter()`

Examples

```
fit_a <- lm(Sepal.Length ~ . -Petal.Length, data = iris)
fit_b <- lm(Sepal.Length ~ ., data = iris)

fl_a <- flashlight(model = fit_a, label = "no Petal.Length")
fl_b <- flashlight(model = fit_b, label = "all")
fls <- multiflashlight(list(fl_a, fl_b), data = iris, y = "Sepal.Length")

plot(light_scatter(fls, v = "Petal.Width"), color = "darkred")

sc <- light_scatter(fls, "Petal.Length", by = "Species", type = "residual")
plot(sc)
```

`most_important` *Most Important Variables.*

Description

Returns the most important variable names sorted descendingly.

Usage

```
most_important(x, top_m = Inf)
```

Arguments

`x`	An object of class "light_importance".
`top_m`	Maximum number of important variables to be returned.

Value

A character vector of variable names sorted in descending importance.

See Also

light_importance()

Examples

```
fit <- lm(Sepal.Length ~ ., data = iris)
fl <- flashlight(model = fit, label = "lm", data = iris, y = "Sepal.Length")
imp <- light_importance(fl)
most_important(imp)
most_important(imp, top_m = 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" (a named list of flashlight objects).

Methods (by class)

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

---

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, rotate_x = FALSE, ...)
```

Arguments

x	An object of class "light_breakdown".
facet_scales	Scales argument passed to `ggplot2::facet_wrap()`.
facet_ncol	ncol argument passed to `ggplot2::facet_wrap()`.
rotate_x	Should x axis labels be rotated by 45 degrees?
...	Further arguments passed to `ggplot2::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 "ggplot".

See Also

light_breakdown()

---

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,
  show_points = TRUE,
  recode_labels = NULL,
  ...
)
```

Arguments

x	An object of class "light_effects".
use	A vector of elements to show. Any subset of ("response", "predicted", "pd", "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/linewidth in ggplot2::geom_point() and ggplot2::geom_line().
facet_scales	Scales argument passed to ggplot2::facet_wrap().
facet_nrow	Number of rows in ggplot2::facet_wrap().
rotate_x	Should x axis labels be rotated by 45 degrees?
show_points	Should points be added to the line (default is TRUE).
recode_labels	Named vector of curve labels. The names refer to the usual labels, while the values are the desired labels, e.g., 'c("partial dependence" = PDP", "ale" = "ALE").
...	Further arguments passed to geoms.

Value

An object of class "ggplot".

See Also

light_effects(), plot_counts()

---

plot.light_global_surrogate

Plot Global Surrogate Trees

Description

Use rpart.plot::rpart.plot() to visualize trees fitted by light_global_surrogate().

Usage

```
## S3 method for class 'light_global_surrogate'
plot(x, type = 5, auto_main = TRUE, mfrow = NULL, ...)
```

Arguments

x	An object of class "light_global_surrogate".
type	Plot type, see help of rpart.plot::rpart.plot(). Default is 5.
auto_main	Automatic plot titles (only if multiple trees are shown).
mfrow	If multiple trees are shown in the same figure: what value of mfrow to use in graphics::par()?
...	Further arguments passed to rpart.plot::rpart.plot().

Value

An object of class "ggplot".

See Also

light_global_surrogate()

plot.light_ice *Visualize ICE profiles*

Description

Minimal visualization of an object of class "light_ice" as `ggplot2::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 `ggplot2::facet_wrap()`.
`rotate_x`	Should x axis labels be rotated by 45 degrees?
`...`	Further arguments passed to `ggplot2::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 "ggplot".

See Also

`light_ice()`

plot.light_importance *Visualize Variable Importance*

Description

Visualization of an object of class "light_importance" via `ggplot2::geom_bar()`. If available, standard errors are added by `ggplot2::geom_errorbar()`. 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",
  rotate_x = FALSE,
  error_bars = TRUE,
  ...
)
```

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 ggplot2::facet_wrap().
rotate_x	Should x axis labels be rotated by 45 degrees?
error_bars	Should error bars be added? Defaults to TRUE. Only available if light_importance() was run with multiple permutations by setting m_repetitions > 1.
...	Further arguments passed to ggplot2::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 "ggplot".

See Also

light_importance()

plot.light_performance *Visualize Model Performance*

Description

Minimal visualization of an object of class "light_performance" as `ggplot2::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,
  geom = c("bar", "point"),
  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.
geom	Geometry of plot (either "bar" or "point")
facet_scales	Scales argument passed to `ggplot2::facet_wrap()`.
rotate_x	Should x axis labels be rotated by 45 degrees?
...	Further arguments passed to `ggplot2::geom_bar()` or `ggplot2::geom_point()`.

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 "ggplot".

See Also

light_performance()

---

plot.light_profile *Visualize Profiles, e.g. 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",
  show_points = TRUE,
  ...
)
```

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 ggplot2::facet_wrap().
rotate_x	Should x axis labels be rotated by 45 degrees?
show_points	Should points be added to the line (default is TRUE).
...	Further arguments passed to ggplot2::geom_point() or ggplot2::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 represented 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, while the second one or the multiflashlight is shown via facet (change with swap_dim).

Value

An object of class "ggplot".

See Also

light_profile(), plot.light_effects()

---

plot.light_profile2d *Visualize 2D-Profiles, e.g., of Partial Dependence*

Description

Minimal visualization of an object of class "light_profile2d". The object returned is of class "ggplot" and can be further customized.

Usage

```
## S3 method for class 'light_profile2d'
plot(x, swap_dim = FALSE, rotate_x = TRUE, numeric_as_factor = FALSE, ...)
```

Arguments

x	An object of class "light_profile2d".
swap_dim	Swap the ggplot2::facet_grid() dimensions.
rotate_x	Should the x axis labels be rotated by 45 degrees? Default is TRUE.
numeric_as_factor	Should numeric x and y values be converted to factors first? Default is FALSE. Useful if cut_type was not set to "equal".
...	Further arguments passed to ggplot2::geom_tile().

Details

The main geometry is ggplot2::geom_tile(). Additional dimensions ("by" variable(s) and/or multiflashlight) are represented by facet_wrap/grid. For all types of profiles except "partial dependence", it is natural to see empty parts in the plot. These are combinations of the v variables that do not appear in the data. Even for type "partial dependence", such gaps can occur, e.g. for cut_type = "quantile" or if n_bins are larger than the number of distinct values of a v variable. Such gaps can be suppressed by setting numeric_as_factor = TRUE or by using the arguments breaks, pd_evaluate_at or pd_grid in light_profile2d().

Value

An object of class "ggplot".

See Also

light_profile2d()

plot.light_scatter *Scatter Plot*

Description

Values are plotted against a variable. The object returned is of class "ggplot" and can be further customized. To avoid overplotting, try `alpha = 0.2` or `position = "jitter"`.

Usage

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

Arguments

`x`	An object of class "light_scatter".
`swap_dim`	If multiflashlight and one "by" variable, or single flashlight with two "by" variables, swap the role of color variable and facet variable. If multiflashlight or one "by" variable, use colors instead of facets.
`facet_scales`	Scales argument passed to `ggplot2::facet_wrap()`.
`rotate_x`	Should x axis labels be rotated by 45 degrees?
`...`	Further arguments passed to `ggplot2::geom_point()`. Typical arguments would be `alpha = 0.2` or `position = "jitter"` to avoid overplotting.

Value

An object of class "ggplot".

See Also

`light_scatter()`

plot_counts *DEPRECATED*

Description

DEPRECATED

Usage

```
plot_counts(...)
```

Arguments

...	Any input.

Value

Error message.

`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.multiflashlight *Predictions for multiflashlight*

Description

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

Usage

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

Arguments

`object`	An object of class "multiflashlight".
`...`	Arguments used to update the multiflashlight.

Value

A named list of prediction vectors.

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")
mod_part <- flashlight(model = fit_part, label = "part")
mods <- multiflashlight(list(mod_full, mod_part), data = iris, y = "Sepal.Length")
predict(mods, data = iris[1:5, ])
```

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

---

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")
```

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)
x <- flashlight(model = fit, data = iris, y = "Sepal.Length", label = "ols")
residuals(x)[1:5]
```

---

residuals.multiflashlight

Residuals for multiflashlight

Description

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

Usage

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

Arguments

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

Value

A named list with residuals per flashlight.

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")
mod_part <- flashlight(model = fit_part, label = "part")
mods <- multiflashlight(list(mod_full, mod_part), data = iris, y = "Sepal.Length")
residuals(mods, data = head(iris))
```

`response` *Response of multi/-flashlight*

Description

Extracts response from object of class "flashlight".

Usage

```
response(object, ...)

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

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

## S3 method for class 'multiflashlight'
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)

- `response(default)`: Default method not implemented yet.
- `response(flashlight)`: Extract response from flashlight object.
- `response(multiflashlight)`: Extract responses from multiflashlight 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