Package ‘shapviz’

August 21, 2024

**Title**  SHAP Visualizations

**Version**  0.9.4

**Description**  Visualizations for SHAP (SHapley Additive exPlanations), such as waterfall plots, force plots, various types of importance plots, dependence plots, and interaction plots. These plots act on a 'shapviz' object created from a matrix of SHAP values and a corresponding feature dataset. Wrappers for the R packages 'xgboost', 'lightgbm', 'fastshap', 'shapr', 'h2o', 'treeshap', 'DALEX', and 'kernelshap' are added for convenience. By separating visualization and computation, it is possible to display factor variables in graphs, even if the SHAP values are calculated by a model that requires numerical features. The plots are inspired by those provided by the 'shap' package in Python, but there is no dependency on it.

**License**  GPL (>= 2)

**Depends**  R (>= 3.6.0)

**Encoding**  UTF-8

**RoxygenNote**  7.3.2

**Imports**  ggfittext (>= 0.8.0), gggenes, ggplot2 (>= 3.4.0), ggrepel, grid, patchwork, rlang (>= 0.3.0), stats, utils, xgboost

**Enhances**  fastshap, h2o, lightgbm

**LazyData**  true

**Suggests**  knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder**  knitr

**Config/testthat/edition**  3

**URL**  https://github.com/ModelOriented/shapviz,
         https://modeloriented.github.io/shapviz/

**BugReports**  https://github.com/ModelOriented/shapviz/issues

**NeedsCompilation**  no

**Author**  Michael Mayer [aut, cre],
             Adrian Stando [ctb]
## Description

Visualizations for SHAP (SHapley Additive exPlanations), such as waterfall plots, force plots, various types of importance plots, dependence plots, and interaction plots. These plots act on a `shapviz` object created from a matrix of SHAP values and a corresponding feature dataset. Wrappers for the R packages `xgboost`, `lightgbm`, `fastshap`, `shapr`, `h2o`, `treeshap`, `DALEX`, and `kernelshap` are added for convenience. By separating visualization and computation, it is possible to display

### Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>shapviz-package</td>
<td>2</td>
</tr>
<tr>
<td>.shapviz</td>
<td>3</td>
</tr>
<tr>
<td>c.shapviz</td>
<td>4</td>
</tr>
<tr>
<td>collapse_shap</td>
<td>5</td>
</tr>
<tr>
<td>dim.shapviz</td>
<td>6</td>
</tr>
<tr>
<td>dimnames.shapviz</td>
<td>6</td>
</tr>
<tr>
<td>dimnames&lt;-.shapviz</td>
<td>7</td>
</tr>
<tr>
<td>extractors</td>
<td>8</td>
</tr>
<tr>
<td>format_max</td>
<td>9</td>
</tr>
<tr>
<td>is.mshapviz</td>
<td>10</td>
</tr>
<tr>
<td>is.shapviz</td>
<td>11</td>
</tr>
<tr>
<td>miami</td>
<td>12</td>
</tr>
<tr>
<td>mshapviz</td>
<td>13</td>
</tr>
<tr>
<td>potential_interactions</td>
<td>13</td>
</tr>
<tr>
<td>print.mshapviz</td>
<td>15</td>
</tr>
<tr>
<td>print.shapviz</td>
<td>15</td>
</tr>
<tr>
<td>rbind.shapviz</td>
<td>16</td>
</tr>
<tr>
<td>shapviz</td>
<td>17</td>
</tr>
<tr>
<td>split.shapviz</td>
<td>22</td>
</tr>
<tr>
<td>summary.shapviz</td>
<td>23</td>
</tr>
<tr>
<td>sv_dependence</td>
<td>23</td>
</tr>
<tr>
<td>sv_dependence2D</td>
<td>26</td>
</tr>
<tr>
<td>sv_force</td>
<td>28</td>
</tr>
<tr>
<td>sv_importance</td>
<td>30</td>
</tr>
<tr>
<td>sv_interaction</td>
<td>33</td>
</tr>
<tr>
<td>sv_waterfall</td>
<td>35</td>
</tr>
<tr>
<td>[.shapviz</td>
<td>37</td>
</tr>
</tbody>
</table>

### Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>shapviz-package</td>
<td>39</td>
</tr>
<tr>
<td>shapviz: SHAP Visualizations</td>
<td></td>
</tr>
</tbody>
</table>
factor variables in graphs, even if the SHAP values are calculated by a model that requires numerical features. The plots are inspired by those provided by the 'shap' package in Python, but there is no dependency on it.

Author(s)

Maintainer: Michael Mayer <mayermichael79@gmail.com>
Other contributors:

- Adrian Stando <adrian.j.stando@gmail.com> [contributor]

See Also

Useful links:

- https://github.com/ModelOriented/shapviz
- https://modeloriented.github.io/shapviz/
- Report bugs at https://github.com/ModelOriented/shapviz/issues

Description

Rowbinds two "shapviz" objects using +.

Usage

```r
## S3 method for class 'shapviz' 
e1 + e2
## S3 method for class 'mshapviz'
e1 + e2
```

Arguments

- `e1` The first object of class "shapviz".
- `e2` The second object of class "shapviz".

Value

A new object of class "shapviz".

See Also

`shapviz()`, `rbind.shapviz()`
Examples

S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1]
s2 <- shapviz(S, X, baseline = 4)[2]
s <- s1 + s2
s

# mshapviz
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1L]
s2 <- shapviz(S, X, baseline = 4)[2L]
s <- mshapviz(c(shp1 = s1, shp2 = s2))
s + s

---

c.shapviz

Concatenates "shapviz" Objects

Description

This function combines two or more (usually named) "shapviz" objects to an object of class "mshapviz".

Usage

## S3 method for class 'shapviz'
c(...)

Arguments

... Any number of (optionally named) "shapviz" objects.

Value

A "mshapviz" object.

See Also

mshapviz()

Examples

S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1]
s2 <- shapviz(S, X, baseline = 4)[2]
s <- c(shp1 = s1, shp2 = s2)
s
Description

This function sums up SHAP values (or SHAP interaction values) of feature groups. Typical application: SHAP values have been generated by a model with one or multiple one-hot encoded variables, but the explanations should be done using the original factor.

Usage

collapse_shap(S, collapse = NULL, ...)

Arguments

S  Either a (n x p) matrix of SHAP values or a (n x p x p) array of SHAP interaction values.
collapse  A named list of character vectors. Each vector specifies the feature names whose SHAP values need to be summed up. The names determine the resulting collapsed column/dimension names.
...
    Currently unused.

Value

A matrix of SHAP values, or an array of SHAP interaction values.

Examples

S <- cbind(
x = c(0.1, 0.1, 0.1),
`age low` = c(0.2, -0.1, 0.1),
`age mid` = c(0, 0.2, -0.2),
`age high` = c(1, -1, 0)
)
collapse <- list(age = c("age low", "age mid", "age high"))
collapse_shap(S, collapse)

# Arrays (as with SHAP interactions)
S_inter <- array(1, dim = c(2, 4, 4), dimnames = list(NULL, letters[1:4], letters[1:4]))
collapse_shap(S_inter, collapse = list(cd = c("c", "d"), ab = c("a", "b")))
## dim.shapviz: Dimensions of "shapviz" Object

### Description
Dimensions of "shapviz" Object

### Usage
```
## S3 method for class 'shapviz'
dim(x)
```

### Arguments
- `x` An object of class "shapviz".

### Value
A numeric vector of length two providing the number of rows and columns of the SHAP matrix stored in `x`.

### See Also
- `shapviz()`

### Examples
```
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
x <- shapviz(S, X)
dim(x)
nrow(x)
ncol(x)
```

## dimnames.shapviz: Dimnames of "shapviz" Object

### Description
This implies to use `colnames(x)` to get the column names of the SHAP and feature matrix (and optional SHAP interaction values).

### Usage
```
## S3 method for class 'shapviz'
dimnames(x)
```

### Examples
```
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
x <- shapviz(S, X)
dimnames(x)
```
Arguments

x  An object of class "shapviz".

Value

Dimnames of the SHAP matrix.

See Also

shapviz()

Examples

S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
x <- shapviz(S, X, baseline = 4)
dimnames(x)
colnames(x)
Examples

S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
x <- shapviz(S, X, baseline = 4)
dimnames(x) <- list(1:2, c("a", "b"))
dimnames(x)
colnames(x) <- c("x", "y")
colnames(x)

extractors

<table>
<thead>
<tr>
<th>extractor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_shap_values</td>
<td>Functions to extract SHAP values, feature values, the baseline, or SHAP interactions from a &quot;(m)shapviz&quot; object.</td>
</tr>
</tbody>
</table>

Usage

get_shap_values(object, ...)

## S3 method for class 'shapviz'
get_shap_values(object, ...)

## S3 method for class 'mshapviz'
get_shap_values(object, ...)

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

get_feature_values(object, ...)

## S3 method for class 'shapviz'
get_feature_values(object, ...)

## S3 method for class 'mshapviz'
get_feature_values(object, ...)

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

get_baseline(object, ...)

## S3 method for class 'shapviz'
get_baseline(object, ...)

## S3 method for class 'mshapviz'
get_baseline(object, ...)

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

get_shap_interactions(object, ...)

## S3 method for class 'shapviz'
get_shap_interactions(object, ...)

## S3 method for class 'mshapviz'
get_shap_interactions(object, ...)

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

Arguments

  object Object to extract something.
  ... Currently unused.

Value

- get_shap_values() returns the matrix of SHAP values,
- get_feature_values() the data.frame of feature values,
- get_baseline() the numeric baseline value, and
- get_shap_interactions() the SHAP interactions of the input.

For objects of class "mshapviz", these functions return lists of those elements.

Examples

S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
shp <- shapviz(S, X, baseline = 4)
get_shap_values(shp)

format_max

---

### Description

Formats a numeric vector in a way that its largest absolute value determines the number of digits after the decimal separator. This function is helpful in perfectly aligning numbers on plots. Does not use scientific formatting.
Usage

format_max(x, digits = 4L, ...)

Arguments

x
A numeric vector to be formatted.
digits
Number of significant digits of the largest absolute value.
...
Further arguments passed to format(), e.g., big.mark = "".

Value

A character vector of formatted numbers.

Examples

x <- c(100, 1, 0.1)
format_max(x)

y <- c(100, 1.01)
format_max(y)
format_max(y, digits = 5)

is.mshapviz

Description

Is object of class "mshapviz"?

Usage

is.mshapviz(object)

Arguments

object
An R object.

Value

Returns TRUE if object has "mshapviz" among its classes, and FALSE otherwise.

See Also

mshapviz()
Examples

```r
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1]
s2 <- shapviz(S, X, baseline = 4)
x <- c(s1 = s1, s2 = s2)
is.mshapviz(x)
is.mshapviz(s1)
```

Description

Is object of class "shapviz"?

Usage

```r
is.shapviz(object)
```

Arguments

- `object`: An R object.

Value

Returns TRUE if object has "shapviz" among its classes, and FALSE otherwise.

See Also

`shapviz()`

Examples

```r
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
shp <- shapviz(S, X)
is.shapviz(shp)
is.shapviz("a")
```
Description

The dataset contains information on 13,932 single-family homes sold in Miami-Dade County in 2016. Besides publicly available information, the dataset creator Steven C. Bourassa has added distance variables, aviation noise as well as latitude and longitude.

More information can be found open-access on https://www.mdpi.com/1595920.

The dataset can also be downloaded via miami <- OpenML::getOMLDataSet(43093)$data.

Usage

miami

Format

A data frame with 13,932 rows and 17 columns:

- PARCELNO  unique identifier for each property. About 1% appear multiple times.
- SALE_PRC  sale price ($)
- LND_SQFOOT  land area (square feet)
- TOT_LVG_AREA  floor area (square feet)
- SPEC_FEAT_VAL  value of special features (e.g., swimming pools) ($)
- RAIL_DIST  distance to the nearest rail line (an indicator of noise) (feet)
- OCEAN_DIST  distance to the ocean (feet)
- WATER_DIST  distance to the nearest body of water (feet)
- CNTR_DIST  distance to the Miami central business district (feet)
- SUBCNTR_DI  distance to the nearest subcenter (feet)
- HWY_DIST  distance to the nearest highway (an indicator of noise) (feet)
- age  age of the structure
- avno60plus  dummy variable for airplane noise exceeding an acceptable level
- structure_quality  quality of the structure
- month_sold  sale month in 2016 (1 = jan)
- LATITUDE, LONGITUDE  Coordinates
mshapviz

Combines compatible "shapviz" Objects

Description
This function combines a list of compatible "shapviz" objects to an object of class "mshapviz". The elements can be named.

Usage
mshapviz(object, ...)

Arguments
- object: List of "shapviz" objects to be concatenated.
- ...: Not used.

Value
A "mshapviz" object.

Examples
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1L]
s2 <- shapviz(S, X, baseline = 4)[2L]
s <- mshapviz(c(shp1 = s1, shp2 = s2))
s

potential_interactions

Interaction Strength

Description
Returns a vector of interaction strengths between variable v and all other variables, see Details.

Usage
potential_interactions(
  obj,
  v,
  nbins = NULL,
  color_num = TRUE,
  scale = FALSE,
  adjusted = FALSE
)
Arguments

**obj**  
An object of class "shapviz".

**v**  
Variable name to calculate potential SHAP interactions for.

**nbins**  
Into how many quantile bins should a numeric v be binned? The default NULL equals the smaller of \( n/20 \) and \( \sqrt{n} \) (rounded up), where \( n \) is the sample size. Ignored if obj contains SHAP interactions.

**color_num**  
Should other ("color") features v’ be converted to numeric, even if they are factors/characters? Default is TRUE. Ignored if obj contains SHAP interactions.

**scale**  
Should adjusted R-squared be multiplied with the sample variance of within-bin SHAP values? If TRUE, bins with stronger vertical scatter will get higher weight. The default is FALSE. Ignored if obj contains SHAP interactions.

**adjusted**  
Should adjusted R-squared be used? Default is FALSE.

Details

If SHAP interaction values are available, the interaction strength between feature v and another feature v’ is measured by twice their mean absolute SHAP interaction values.

Otherwise, we use a heuristic calculated as follows:

1. If v is numeric, it is binned into nbins bins.
2. Per bin, the SHAP values of v are regressed onto v, and the R-squared is calculated. Rows with missing v’ are discarded.
3. The R-squared are averaged over bins, weighted by the number of non-missing v’ values.

This measures how much variability in the SHAP values of v is explained by v’, after accounting for v.

Set scale = TRUE to multiply the R-squared by the within-bin variance of the SHAP values. This will put higher weight to bins with larger scatter.

Set color_num = FALSE to not turn the values of the "color" feature v’ to numeric.

Finally, set adjusted = TRUE to use adjusted R-squared.

The algorithm does not consider observations with missing v’ values.

Value

A named vector of decreasing interaction strengths.

See Also

`sv_dependence()`
print.mshapviz

Prints "mshapviz" Object

Description
Prints "mshapviz" Object

Usage
## S3 method for class 'mshapviz'
print(x, ...)

Arguments
x
An object of class "mshapviz".
...
Further arguments passed from other methods.

Value
Invisibly, the input is returned.

See Also
mshapviz()

Examples
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1]
s2 <- shapviz(S, X, baseline = 4)
x <- c(s1 = s1, s2 = s2)
x

print.shapviz

Prints "shapviz" Object

Description
Prints "shapviz" Object

Usage
## S3 method for class 'shapviz'
print(x, ...)

Examples
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1]
s2 <- shapviz(S, X, baseline = 4)
x <- c(s1 = s1, s2 = s2)
x
### Arguments

- **x**: An object of class "shapviz".
- **...**: Further arguments passed from other methods.

### Value

Invisibly, the input is returned.

### See Also

- `shapviz()`

### Examples

```r
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
x <- shapviz(S, X, baseline = 4)
x
```

---

### Description

Rowbinds multiple "shapviz" objects based on the + operator.

### Usage

```r
## S3 method for class 'shapviz'
rbind(...)  
## S3 method for class 'mshapviz'
rbind(...)  
```

### Arguments

- **...**: Any number of "shapviz" or "mshapviz" objects.

### Value

A new object of class "shapviz" or "mshapviz".

### See Also

- `shapviz()`, `mshapviz()`
Examples

```r
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1]
s2 <- shapviz(S, X, baseline = 4)[2]
s <- rbind(s1, s2)
s
# mshapviz
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
s1 <- shapviz(S, X, baseline = 4)[1L]
s2 <- shapviz(S, X, baseline = 4)[2L]
s <- mshapviz(c(shp1 = s1, shp2 = s2))
rbind(s, s)
```

shapviz  

Initialize "shapviz" Object

Description

This function creates an object of class "shapviz" from a matrix of SHAP values, or from a fitted model of type

- XGBoost,
- LightGBM, or
- H2O (tree-based regression or binary classification model).

Furthermore, shapviz() can digest the results of

- fastshap::explain(),
- shapr::explain(),
- treeshap::treeshap(),
- DALEX::predict_parts(),
- kernelshap::kernelshap(),
- kernelshap::permshap(), and
- kernelshap::additive_shap(),

check the vignettes for examples.
Usage

shapviz(object, ...)

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

## S3 method for class 'matrix'
shapviz(object, X, baseline = 0, collapse = NULL, S_inter = NULL, ...)

## S3 method for class 'xgb.Booster'
shapviz(
  object,
  X_pred,
  X = X_pred,
  which_class = NULL,
  collapse = NULL,
  interactions = FALSE,
  ...
)

## S3 method for class 'lgb.Booster'
shapviz(object, X_pred, X = X_pred, which_class = NULL, collapse = NULL, ...)

## S3 method for class 'explain'
shapviz(object, X = NULL, baseline = NULL, collapse = NULL, ...)

## S3 method for class 'treeshap'
shapviz(
  object,
  X = object[['observations']],
  baseline = 0,
  collapse = NULL,
  ...
)

## S3 method for class 'predict_parts'
shapviz(object, ...)

## S3 method for class 'shapr'
shapviz(object, X = object[['x_test']], collapse = NULL, ...)

## S3 method for class 'kernelshap'
shapviz(object, X = object[['X']], which_class = NULL, collapse = NULL, ...)

## S3 method for class 'H2ORegressionModel'
shapviz(object, X_pred, X = as.data.frame(X_pred), collapse = NULL, ...)

## S3 method for class 'H2OBinomialModel'
shapviz(object, X_pred, X = as.data.frame(X_pred), collapse = NULL, ...)

## S3 method for class 'H2OModel'
shapviz(object, X_pred, X = as.data.frame(X_pred), collapse = NULL, ...)

**Arguments**

- **object**
  For XGBoost, LightGBM, and H2O, this is the fitted model used to calculate SHAP values from X_pred. In the other cases, it is the object containing the SHAP values.

- **...**
  Parameters passed to other methods (currently only used by the predict() functions of XGBoost, LightGBM, and H2O).

- **X**
  Matrix or data.frame of feature values used for visualization. Must contain at least the same column names as the SHAP matrix represented by object/X_pred (after optionally collapsing some of the SHAP columns).

- **baseline**
  Optional baseline value, representing the average response at the scale of the SHAP values. It will be used for plot methods that explain single predictions.

- **collapse**
  A named list of character vectors. Each vector specifies the feature names whose SHAP values need to be summed up. The names determine the resulting collapsed column/dimension names.

- **S_inter**
  Optional 3D array of SHAP interaction values. If object has shape n x p, then S_inter needs to be of shape n x p x p. Summation over the second (or third) dimension should yield the usual SHAP values. Furthermore, dimensions 2 and 3 are expected to be symmetric. Default is NULL.

- **X_pred**
  Data set as expected by the predict() function of XGBoost, LightGBM, or H2O. For XGBoost, a matrix or xgb.DMatrix, for LightGBM a matrix, and for H2O a data.frame or an H2OFrame. Only used for XGBoost, LightGBM, or H2O objects.

- **which_class**
  In case of a multiclass or multioutput setting, which class/output (>= 1) to explain. Currently relevant for XGBoost, LightGBM, kernelshap, and permshap.

- **interactions**
  Should SHAP interactions be calculated (default is FALSE)? Only available for XGBoost.

**Details**

Together with the main input, a data set X of feature values is required, used only for visualization. It can therefore contain character or factor variables, even if the SHAP values were calculated from a purely numerical feature matrix. In addition, to improve visualization, it can sometimes be useful to truncate gross outliers, logarithmize certain columns, or replace missing values with an explicit value.

SHAP values of dummy variables can be combined using the convenient collapse argument. Multi-output models created from XGBoost, LightGBM, "kernelshap", or "permshap" return a "mshapviz" object, containing a "shapviz" object per output.
Value

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

- **S**: Numeric matrix of SHAP values.
- **X**: `data.frame` containing the feature values corresponding to S.
- **baseline**: Baseline value, representing the average prediction at the scale of the SHAP values.
- **S_inter**: Numeric array of SHAP interaction values (or NULL).

Methods (by class)

- `shapviz(default)`: Default method to initialize a "shapviz" object.
- `shapviz(matrix)`: Creates a "shapviz" object from a matrix of SHAP values.
- `shapviz(xgb.Booster)`: Creates a "shapviz" object from an XGBoost model.
- `shapviz(lgb.Booster)`: Creates a "shapviz" object from a LightGBM model.
- `shapviz(explain)`: Creates a "shapviz" object from `fastshap::explain()`.
- `shapviz(treeshap)`: Creates a "shapviz" object from `treeshap::treeshap()`.
- `shapviz(predict_parts)`: Creates a "shapviz" object from `DALEX::predict_parts()`.
- `shapviz(shapr)`: Creates a "shapviz" object from `shapr::explain()`.
- `shapviz(kernelshap)`: Creates a "shapviz" object from `kernelshap()`, `permshap()`, and `additive_shap()`.
- `shapviz(H2ORegressionModel)`: Creates a "shapviz" object from a (tree-based) H2O regression model.
- `shapviz(H2OBinomialModel)`: Creates a "shapviz" object from a (tree-based) H2O binary classification model.
- `shapviz(H2OModel)`: Creates a "shapviz" object from a (tree-based) H2O model (base class).

See Also

`sv_importance()`, `sv_dependence()`, `sv_dependence2D()`, `sv_interaction()`, `sv_waterfall()`, `sv_force()`, `collapse_shap()`

Examples

```r
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
shapviz(S, X, baseline = 4)
# XGBoost models
X_pred <- data.matrix(iris[, -1])
dtrain <- xgboost::xgb.DMatrix(X_pred, label = iris[, 1], nthread = 1)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)

# Will use numeric matrix "X_pred" as feature matrix
X <- shapviz(fit, X_pred = X_pred)
X
sv_dependence(X, "Species")
```
```r
# Will use original values as feature matrix
x <- shapviz(fit, X_pred = X_pred, X = iris)
sv_dependence(x, "Species")

# "X_pred" can also be passed as xgb.DMatrix, but only if X is passed as well!
x <- shapviz(fit, X_pred = dtrain, X = iris)

# Multiclass setting
params <- list(objective = "multi:softprob", num_class = 3)
X_pred <- data.matrix(iris[, -5])
dtrain <- xgboost::xgb.DMatrix(
  X_pred, label = as.integer(iris[, 5]) - 1, nthread = 1
)
fit <- xgboost::xgb.train(params = params, data = dtrain, nrounds = 10, nthread = 1)

# Select specific class
x <- shapviz(fit, X_pred = X_pred, which_class = 3)
x

# Or combine all classes to "mshapviz" object
x <- shapviz(fit, X_pred = X_pred)
x

# What if we would have one-hot-encoded values and want to explain the original column?
X_pred <- stats::model.matrix(~ . - 1, iris[, -1])
dtrain <- xgboost::xgb.DMatrix(X_pred, label = as.integer(iris[, 1]), nthread = 1)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)
x <- shapviz(fit, X_pred = X_pred, X = iris,
collapse = list(Species = c("Speciessetosa", "Speciesversicolor", "Speciesvirginica"))
)
summary(x)

# Similarly with LightGBM
if (requireNamespace("lightgbm", quietly = TRUE)) {
  fit <- lightgbm::lgb.train(
    params = list(objective = "regression", num_thread = 1),
    data = lightgbm::lgb.Dataset(X_pred, label = iris[, 1]),
    nrounds = 10,
    verbose = -2
  )
x <- shapviz(fit, X_pred = X_pred)
x

# Multiclass
params <- list(objective = "multiclass", num_class = 3, num_thread = 1)
X_pred <- data.matrix(iris[, -5])
dtrain <- lightgbm::lgb.Dataset(X_pred, label = as.integer(iris[, 5]) - 1)
fit <- lightgbm::lgb.train(params = params, data = dtrain, nrounds = 10)
```

# Select specific class
x <- shapviz(fit, X_pred = X_pred, which_class = 3)
x

# Or combine all classes to a "mshapviz" object
mx <- shapviz(fit, X_pred = X_pred)
mx
all.equal(mx[[3]], x)
}

## Not run:
dtrain <- xgboost::xgb.DMatrix(data.matrix(iris[, -1]), label = iris[, 1])
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)
sv <- shapviz(fit, X_pred = dtrain, X = iris)
mx <- split(sv, f = iris$Species)
sv_dependence(mx, "Petal.Length")
## End(Not run)
**summary.shapviz**

*Summarizes "shapviz" Object*

**Description**

Summarizes "shapviz" Object

**Usage**

```r
## S3 method for class 'shapviz'
summary(object, n = 2L, ...)
```

**Arguments**

- `object`: An object of class "shapviz".
- `n`: Maximum number of rows of SHAP values and feature values to show.
- `...`: Further arguments passed from other methods.

**Value**

Invisibly, the input is returned.

**See Also**

`shapviz()`

**Examples**

```r
S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
object <- shapviz(S, X, baseline = 4)
summary(object)
```

---

**sv_dependence**

*SHAP Dependence Plot*

**Description**

Scatterplot of the SHAP values of a feature against its feature values. If SHAP interaction values are available, setting `interactions = TRUE` allows to focus on pure interaction effects (multiplied by two) or on pure main effects. By default, the feature on the color scale is selected via SHAP interactions (if available) or an interaction heuristic, see `potential_interactions()`.
sv_dependence

Usage

sv_dependence(object, ...)

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

## S3 method for class 'shapviz'
sv_dependence(
  object,
  v,
  color_var = "auto",
  color = "#3b528b",
  viridis_args = getOption("shapviz.viridis_args"),
  jitter_width = NULL,
  interactions = FALSE,
  ih_nbins = NULL,
  ih_color_num = TRUE,
  ih_scale = FALSE,
  ih_adjusted = FALSE,
  ...
)

## S3 method for class 'mshapviz'
sv_dependence(
  object,
  v,
  color_var = "auto",
  color = "#3b528b",
  viridis_args = getOption("shapviz.viridis_args"),
  jitter_width = NULL,
  interactions = FALSE,
  ih_nbins = NULL,
  ih_color_num = TRUE,
  ih_scale = FALSE,
  ih_adjusted = FALSE,
  ...
)

Arguments

object      An object of class "(m)shapviz".
...
Arguments passed to `ggplot2::geom_jitter()`.
v         Column name of feature to be plotted. Can be a vector/list if object is of class "shapviz".

color_var Feature name to be used on the color scale to investigate interactions. The default ("auto") uses SHAP interaction values (if available), or a heuristic to select the strongest interacting feature. Set to `NULL` to not use the color axis. Can be a vector/list if object is of class "shapviz".
sv_dependence

- **color**: Color to be used if `color_var = NULL`. Can be a vector/list if `v` is a vector.
- **viridis_args**: List of viridis color scale arguments, see `?ggplot2::scale_color_viridis_c`. The default points to the global option `shapviz.viridis_args`, which corresponds to `list(begin = 0.25, end = 0.85, option = "inferno")`. These values are passed to `ggplot2::scale_color_viridis_*()`. For example, to switch to a standard viridis scale, you can either change the default via `options(shapviz.viridis_args = list())`, or set `viridis_args = list()`. Only relevant if `color_var` is not `NULL`.
- **jitter_width**: The amount of horizontal jitter. The default (NULL) will use a value of 0.2 in case `v` is discrete, and no jitter otherwise. (Numeric variables are considered discrete if they have at most 7 unique values.) Can be a vector/list if `v` is a vector.
- **interactions**: Should SHAP interaction values be plotted? Default is `FALSE`. Requires SHAP interaction values. If `color_var = NULL` (or it is equal to `v`), the pure main effect of `v` is visualized. Otherwise, twice the SHAP interaction values between `v` and the `color_var` are plotted.
- **ih_nbins**, **ih_color_num**, **ih_scale**, **ih_adjusted**: Interaction heuristic (ih) parameters used to select the color variable, see `potential_interactions()`. Only used if `color_var = "auto"` and if there are no SHAP interaction values.

**Value**

An object of class "ggplot" (or "patchwork") representing a dependence plot.

**Methods (by class)**

- **sv_dependence(default)**: Default method.
- **sv_dependence(shapviz)**: SHAP dependence plot for "shapviz" object.
- **sv_dependence(mshapviz)**: SHAP dependence plot for "mshapviz" object.

**See Also**

`potential_interactions()`

**Examples**

```r
library(shapviz)

# Setup

# Create data
iris[, -5] <- as.numeric(iris[, -5])
dtrain <- xgboost::xgb.DMatrix(
    data.matrix(iris[, -1]), label = iris[, 1], nthread = 1
)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)
x <- shapviz(fit, X_pred = dtrain, X = iris)
sv_dependence(x, "Petal.Length")
sv_dependence(x, "Petal.Length", color_var = "Species")
sv_dependence(x, "Petal.Length", color_var = NULL)
sv_dependence(x, c("Species", "Petal.Length"))
sv_dependence(x, "Petal.Width", color_var = c("Species", "Petal.Length"))

# SHAP interaction values/main effects
x2 <- shapviz(fit, X_pred = dtrain, X = iris, interactions = TRUE)
sv_dependence(x2, "Petal.Length", interactions = TRUE)
```
sv_dependence2D

sv_dependence(
  x2, c("Petal.Length", "Species"), color_var = NULL, interactions = TRUE
)

---

**Description**

Scatterplot of two features, showing the sum of their SHAP values on the color scale. This allows to visualize the combined effect of two features, including interactions. A typical application are models with latitude and longitude as features (plus maybe other regional features that can be passed via add_vars).

If SHAP interaction values are available, setting interactions = TRUE allows to focus on pure interaction effects (multiplied by two). In this case, add_vars has no effect.

**Usage**

sv_dependence2D(object, ...)

## Default S3 method:
sv_dependence2D(object, ...)

## S3 method for class 'shapviz'
sv_dependence2D(
  object,
  x,
  y,
  viridis_args = getOption("shapviz.viridis_args"),
  jitter_width = NULL,
  jitter_height = NULL,
  interactions = FALSE,
  add_vars = NULL,
  ...
)

## S3 method for class 'mshapviz'
sv_dependence2D(
  object,
  x,
  y,
  viridis_args = getOption("shapviz.viridis_args"),
  jitter_width = NULL,
  jitter_height = NULL,
  interactions = FALSE,
  add_vars = NULL,
  ...
)
sv_dependence2D

Arguments

- **object**: An object of class "(m)shapviz".
- **Arguments passed to `ggplot2::geom_jitter()`**
- **x**: Feature name for x axis. Can be a vector/list if object is of class "shapviz".
- **y**: Feature name for y axis. Can be a vector/list if object is of class "shapviz".
- **viridis_args**: List of viridis color scale arguments, see `?ggplot2::scale_color_viridis_c`. The default points to the global option `shapviz.viridis_args`, which corresponds to `list(begin = 0.25, end = 0.85, option = "inferno")`. These values are passed to `ggplot2::scale_color_viridis_*()`. For example, to switch to a standard viridis scale, you can either change the default via `options(shapviz.viridis_args = list())`, or set `viridis_args = list()`. Only relevant if `color_var` is not `NULL`.
- **jitter_width**: The amount of horizontal jitter. The default (NULL) will use a value of 0.2 in case v is discrete, and no jitter otherwise. (Numeric variables are considered discrete if they have at most 7 unique values.) Can be a vector/list if v is a vector.
- **jitter_height**: Similar to `jitter_width` for vertical scatter.
- **interactions**: Should SHAP interaction values be plotted? The default (FALSE) will show the rowwise sum of the SHAP values of x and y. If TRUE, will use twice the SHAP interaction value (requires SHAP interactions).
- **add_vars**: Optional vector of feature names, whose SHAP values should be added to the sum of the SHAP values of x and y (only if `interactions = FALSE`). A use case would be a model with geographic x and y coordinates, along with some additional locational features like distance to the next train station.

Value

An object of class "ggplot" (or "patchwork") representing a dependence plot.

Methods (by class)

- `sv_dependence2D(default)`: Default method.
- `sv_dependence2D(shapviz)`: 2D SHAP dependence plot for "shapviz" object.
- `sv_dependence2D(mshapviz)`: 2D SHAP dependence plot for "mshapviz" object.

See Also

`sv_dependence()`

Examples

dtrain <- xgboost::xgb.DMatrix(
  data.matrix(iris[, -1]), label = iris[, 1], nthread = 1
)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)
sv <- shapviz(fit, X_pred = dtrain, X = iris)
sv_dependence2D(sv, x = "Petal.Length", y = "Species")
sv_dependence2D(sv, x = c("Petal.Length", "Species"), y = "Sepal.Width")

# SHAP interaction values
sv2 <- shapviz(fit, X_pred = dtrain, X = iris, interactions = TRUE)
sv_dependence2D(sv2, x = "Petal.Length", y = "Species", interactions = TRUE)
sv_dependence2D(
  sv2, x = "Petal.Length", y = c("Species", "Petal.Width"), interactions = TRUE)

# mshapviz object
mx <- split(sv, f = iris$Species)
sv_dependence2D(mx, x = "Petal.Length", y = "Sepal.Width")

sv_force

SHAP Force Plot

Description

Creates a force plot of SHAP values of one observation. If multiple observations are selected, their SHAP values and predictions are averaged.

Usage

sv_force(object, ...)

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

## S3 method for class 'shapviz'
sv_force(
  object,
  row_id = 1L,
  max_display = 6L,
  fill_colors = c("#f7d13d", "#a52c60"),
  format_shap = getOption("shapviz.format_shap"),
  format_feat = getOption("shapviz.format_feat"),
  contrast = TRUE,
  bar_label_size = 3.2,
  show_annotation = TRUE,
  annotation_size = 3.2,
  ...
)

## S3 method for class 'mshapviz'
sv_force(
  object,
  row_id = 1L,
  max_display = 6L,
fill_colors = c("#f7d13d", "#a52c60"),
format_shap = getOption("shapviz.format_shap"),
format_feat = getOption("shapviz.format_feat"),
contrast = TRUE,
bar_label_size = 3.2,
show_annotation = TRUE,
annotation_size = 3.2,
)

Arguments

object

Arguments passed to ggfittext::geom_fit_text(). For example, size = 9 will use fixed text size in the bars and size = 0 will altogether suppress adding text to the bars.
row_id

Subset of observations to plot, typically a single row number. If more than one row is selected, SHAP values are averaged, and feature values are shown only when they are unique.
max_display

Maximum number of features (with largest absolute SHAP values) should be plotted? If there are more features, they will be collapsed to one feature. Set to Inf to show all features.
fill_colors

A vector of exactly two fill colors: the first for positive SHAP values, the other for negative ones.
format_shap

Function used to format SHAP values. The default uses the global option shapviz.format_shap, which equals to function(z) prettyNum(z, digits = 3, scientific = FALSE) by default.
format_feat

Function used to format numeric feature values. The default uses the global option shapviz.format_feat, which equals to function(z) prettyNum(z, digits = 3, scientific = FALSE) by default.
contrast

Logical flag that determines whether to use white text in dark arrows. Default is TRUE.
bar_label_size

Size of text used to describe bars (via ggrepel::geom_text_repel()).
show_annotation

Should "f(x)" and "E(f(x))" be plotted? Default is TRUE.
annotation_size

Size of the annotation text (f(x)=... and E(f(x))=...).

Details

f(x) denotes the prediction on the SHAP scale, while E(f(x)) refers to the baseline SHAP value.

Value

An object of class "ggplot" (or "patchwork") representing a force plot.
### sv_importance

**Description**

This function provides two types of SHAP importance plots: a bar plot and a beeswarm plot (sometimes called "SHAP summary plot"). The two types of plots can also be combined.

**Usage**

```r
sv_importance(object, ...)  
```

## Default S3 method:

```r
sv_importance(object, ...)  
```

## S3 method for class 'shapviz'

```r
sv_importance(object, 
              kind = c("bar", "beeswarm", "both", "no"),
              max_display = 15L,
              fill = "#fca50a",
              bar_width = 2/3,
              bee_width = 0.4,
              bee_adjust = 0.5,
              viridis_args = getOption("shapviz.viridis_args"),
```
sv_importance

## S3 method for class 'mshapviz'
sv_importance(
  object,
  kind = c("bar", "beeswarm", "both", "no"),
  max_display = 15L,
  fill = "#fca50a",
  bar_width = 2/3,
  bar_type = c("dodge", "stack", "facets", "separate"),
  bee_width = 0.4,
  bee_adjust = 0.5,
  viridis_args = getOption("shapviz.viridis_args"),
  color_bar_title = "Feature value",
  show_numbers = FALSE,
  format_fun = format_max,
  number_size = 3.2,
  sort_features = TRUE,
  ...
)

Arguments

object
An object of class "(m)shapviz".

... Arguments passed to ggplot2::geom_bar() (if kind = "bar") or to ggplot2::geom_point() otherwise. For instance, passing alpha = 0.2 will produce semi-transparent beeswarms, and setting size = 3 will produce larger dots.

kind
Should a "bar" plot (the default), a "beeswarm" plot, or "both" be shown? Set to "no" in order to suppress plotting. In that case, the sorted SHAP feature importances of all variables are returned.

max_display
How many features should be plotted? Set to Inf to show all features. Has no effect if kind = "no".

fill
Color used to fill the bars (only used if bars are shown).

bar_width
Relative width of the bars (only used if bars are shown).

bee_width
Relative width of the beeswarms.

bee_adjust
Relative bandwidth adjustment factor used in estimating the density of the beeswarms.

viridis_args
List of viridis color scale arguments. The default points to the global option shapviz.viridis_args, which corresponds to list(begin = 0.25, end = 0.85, option = "inferno"). These values are passed to ggplot2::scale_color_viridis_c(). For example, to switch to standard viridis, either change the default with options(shapviz.viridis_args = list()) or set viridis_args = list().
color_bar_title
   Title of color bar of the beeswarm plot. Set to NULL to hide the color bar altogether.

show_numbers
   Should SHAP feature importances be printed? Default is FALSE.

format_fun
   Function used to format SHAP feature importances (only if show_numbers = TRUE). To change to scientific notation, use function(x) = prettyNum(x, scientific = TRUE).

number_size
   Text size of the numbers (if show_numbers = TRUE).

sort_features
   Should features be sorted or not? The default is TRUE.

bar_type
   For "mshapviz" objects with kind = "bar": How should bars be represented? The default is "dodge" for dodged bars. Other options are "stack", "wrap", or "separate" (via "patchwork"). Note that "separate" is currently the only option that supports show_numbers = TRUE.

Details

The bar plot shows SHAP feature importances, calculated as the average absolute SHAP value per feature. The beeswarm plot displays SHAP values per feature, using min-max scaled feature values on the color axis. Non-numeric features are transformed to numeric by calling data.matrix() first. For both types of plots, the features are sorted in decreasing order of importance.

Value

A "ggplot" (or "patchwork") object representing an importance plot, or - if kind = "no" - a named numeric vector of sorted SHAP feature importances (or a matrix in case of an object of class "mshapviz").

Methods (by class)

- sv_importance(default): Default method.
- sv_importance(shapviz): SHAP importance plot for an object of class "shapviz".
- sv_importance(mshapviz): SHAP importance plot for an object of class "mshapviz".

See Also

sv_interaction

Examples

X_train <- data.matrix(iris[, -1])
dtrain <- xgboost::xgb.DMatrix(X_train, label = iris[, 1], nthread = 1)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)
x <- shapviz(fit, X_pred = X_train)
sv_importance(x)
sv_importance(x, kind = "no")
sv_importance(x, kind = "beeswarm", show_numbers = TRUE)
sv_interaction

SHAP Interaction Plot

Description

Plots a beeswarm plot for each feature pair. Diagonals represent the main effects, while off-diagonals show interactions (multiplied by two due to symmetry). The colors on the beeswarm plots represent min-max scaled feature values. Non-numeric features are transformed to numeric by calling `data.matrix()` first. The features are sorted in decreasing order of usual SHAP importance.

Usage

```r
sv_interaction(object, ...)  
## Default S3 method:
sv_interaction(object, ...)  
## S3 method for class 'shapviz'
sv_interaction(
  object,
  kind = c("beeswarm", "no"),
  max_display = 7L,
  alpha = 0.3,
  bee_width = 0.3,
  bee_adjust = 0.5,
  viridis_args = getOption("shapviz.viridis_args"),
  color_bar_title = "Row feature value",
  sort_features = TRUE,
  ...
)
```

```r
## S3 method for class 'mshapviz'
sv_interaction(
  object,
  kind = c("beeswarm", "no"),
  max_display = 7L,
  alpha = 0.3,
  bee_width = 0.3,
  bee_adjust = 0.5,
  viridis_args = getOption("shapviz.viridis_args"),
  color_bar_title = "Row feature value",
  sort_features = TRUE,
  ...
)
```
sv_interaction

Arguments

object An object of class "(m)shapviz" containing element S_inter.

... Arguments passed to ggplot2::geom_point(). For instance, passing size = 1 will produce smaller dots.

kind Set to "no" to return the matrix of average absolute SHAP interactions (or a list of such matrices in case of object of class "mshapviz"). Due to symmetry, off-diagonals are multiplied by two. The default is "beeswarm".

max_display How many features should be plotted? Set to Inf to show all features. Has no effect if kind = "no".

alpha Transparency of the beeswarm dots. Defaults to 0.3.

bee_width Relative width of the beeswarms.

bee_adjust Relative bandwidth adjustment factor used in estimating the density of the beeswarms.

viridis_args List of viridis color scale arguments. The default points to the global option shapviz.viridis_args, which corresponds to list(begin = 0.25, end = 0.85, option = "inferno"). These values are passed to ggplot2::scale_color_viridis_c(). For example, to switch to standard viridis, either change the default with options(shapviz.viridis_args = list()) or set viridis_args = list().

color_bar_title Title of color bar of the beeswarm plot. Set to NULL to hide the color bar altogether.

sort_features Should features be sorted or not? The default is TRUE.

Value

A "ggplot" (or "patchwork") object, or - if kind = "no" - a named numeric matrix of average absolute SHAP interactions sorted by the average absolute SHAP values (or a list of such matrices in case of "mshapviz" object).

Methods (by class)

• sv_interaction(default): Default method.
• sv_interaction(shapviz): SHAP interaction plot for an object of class "shapviz".
• sv_interaction(mshapviz): SHAP interaction plot for an object of class "mshapviz".

See Also

sv_importance()

Examples

dtrain <- xgboost::xgb.DMatrix(
  data.matrix(iris[, -1]), label = iris[, 1], nthread = 1
)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 10, nthread = 1)
x <- shapviz(fit, X_pred = dtrain, X = iris, interactions = TRUE)
sv_interaction(x, kind = "no")
sv_interaction(x, max_display = 2, size = 3)
sv_waterfall

Description

Creates a waterfall plot of SHAP values of one observation. If multiple observations are selected, their SHAP values and predictions are averaged.

Usage

sv_waterfall(object, ...)

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

## S3 method for class 'shapviz'
sv_waterfall(
  object,
  row_id = 1L,
  max_display = 10L,
  order_fun = function(s) order(abs(s)),
  fill_colors = c("#f7d13d", "#a52c60"),
  format_shap = getOption("shapviz.format_shap"),
  format_feat = getOption("shapviz.format_feat"),
  contrast = TRUE,
  show_connection = TRUE,
  show_annotation = TRUE,
  annotation_size = 3.2,
  ...
)

## S3 method for class 'mshapviz'
sv_waterfall(
  object,
  row_id = 1L,
  max_display = 10L,
  order_fun = function(s) order(abs(s)),
  fill_colors = c("#f7d13d", "#a52c60"),
  format_shap = getOption("shapviz.format_shap"),
  format_feat = getOption("shapviz.format_feat"),
  contrast = TRUE,
  show_connection = TRUE,
  show_annotation = TRUE,
  annotation_size = 3.2,
  ...
)
Arguments

object  
An object of class "(m)shapviz".

Arguments passed to `ggfittext::geom_fit_text()`. For example, size = 9 will use fixed text size in the bars and size = 0 will altogether suppress adding text to the bars.

row_id  
Subset of observations to plot, typically a single row number. If more than one row is selected, SHAP values are averaged, and feature values are shown only when they are unique.

max_display  
Maximum number of features (with largest absolute SHAP values) should be plotted? If there are more features, they will be collapsed to one feature. Set to Inf to show all features.

order_fun  
Function specifying the order of the variables/SHAP values. It maps the vector s of SHAP values to sort indices from 1 to length(s). The default is function(s) order(abs(s)). To plot without sorting, use function(s) 1:length(s) or function(s) length(s):1.

fill_colors  
A vector of exactly two fill colors: the first for positive SHAP values, the other for negative ones.

format_shap  
Function used to format SHAP values. The default uses the global option shapviz.format_shap, which equals to function(z) prettyNum(z, digits = 3, scientific = FALSE) by default.

format_feat  
Function used to format numeric feature values. The default uses the global option shapviz.format_feat, which equals to function(z) prettyNum(z, digits = 3, scientific = FALSE) by default.

contrast  
Logical flag that determines whether to use white text in dark arrows. Default is TRUE.

show_connection  
Should connecting lines be shown? Default is TRUE.

show_annotation  
Should "f(x)" and "E(f(x))" be plotted? Default is TRUE.

annotation_size  
Size of the annotation text (f(x)=... and E(f(x))=...).

Details

f(x) denotes the prediction on the SHAP scale, while E(f(x)) refers to the baseline SHAP value.

Value

An object of class "ggplot" (or "patchwork") representing a waterfall plot.

Methods (by class)

- `sv_waterfall(default)`: Default method.
- `sv_waterfall(shapviz)`: SHAP waterfall plot for an object of class "shapviz".
- `sv_waterfall(mshapviz)`: SHAP waterfall plot for an object of class "mshapviz".
See Also

sv_force()

Examples

dtrain <- xgboost::xgb.DMatrix(
  data.matrix(iris[, -1]), label = iris[, 1], nthread = 1)
fit <- xgboost::xgb.train(data = dtrain, nrounds = 20, nthread = 1)
x <- shapviz(fit, X_pred = dtrain, X = iris[, -1])
sv_waterfall(x)
sv_waterfall(x, row_id = 123, max_display = 2, size = 9, fill_colors = 4:5)
# Ordered by colnames(x), combined with max_display
sv_waterfall(
  x[, sort(colnames(x))], order_fun = function(s) length(s):1, max_display = 3
)
# Aggregate over all observations with Petal.Length == 1.4
sv_waterfall(x, row_id = x$X$Petal.Length == 1.4)

Description

Use standard square bracket subsetting to select rows and/or columns of SHAP values, feature values, and SHAP interaction values of a "shapviz" object.

Usage

```r
## S3 method for class 'shapviz'
x[i, j, ...]
```

Arguments

- `x` An object of class "shapviz".
- `i` Row subsetting.
- `j` Column subsetting.
- `...` Currently unused.

Value

A new object of class "shapviz".

See Also

shapviz()
Examples

S <- matrix(c(1, -1, -1, 1), ncol = 2, dimnames = list(NULL, c("x", "y")))
X <- data.frame(x = c("a", "b"), y = c(100, 10))
x <- shapviz(S, X, baseline = 4)
x[1, "x"]
x[1]
x[c(FALSE, TRUE), ]
x[, "x"]
Index

* datasets
  miami, 12
  +.mshapviz (+.shapviz), 3
  +.shapviz, 3
  [].shapviz, 37
  c.shapviz, 4
  collapse_shap, 5
  collapse_shap(), 20
  data.matrix(), 32, 33
  dim.shapviz, 6
  dimnames.shapviz, 6
  dimnames<-.shapviz, 7
  extractors, 8
  format(), 10
  format_max, 9
  get_baseline(extractors), 8
  get_feature_values(extractors), 8
  get_shap_interactions(extractors), 8
  get_shap_values(extractors), 8
  ggfittext::geom_fit_text(), 29, 36
  ggplot2::geom_bar(), 31
  ggplot2::geom_jitter(), 24, 27
  ggplot2::geom_point(), 31, 34
  ggplot2::scale_color_viridis_c(), 31, 34
  ggrepel::geom_text_repel(), 29
  is.mshapviz, 10
  is.shapviz, 11
  miami, 12
  mshapviz, 13
  mshapviz(), 4, 10, 15, 16
  potential_interactions, 13
  potential_interactions(), 23, 25
  print.mshapviz, 15
  print.shapviz, 15
  rbind.mshapviz (rbind.shapviz), 16
  rbind.shapviz, 16
  rbind.shapviz(), 3, 22
  shapviz, 17
  shapviz(). 3, 6, 7, 11, 16, 17, 22, 23, 37
  shapviz-package, 2
  split.shapviz, 22
  summary.shapviz, 23
  sv_dependence, 23
  sv_dependence(), 14, 20, 27
  sv_dependence2D, 26
  sv_dependence2D(), 20
  sv_force, 28
  sv_force(), 20, 37
  sv_importance, 30
  sv_importance(), 20, 34
  sv_interaction, 32, 33
  sv_interaction(), 20
  sv_waterfall, 35
  sv_waterfall(), 20, 30
  sv_waterfall()