Package ‘shapviz’

January 11, 2023

Title SHAP Visualizations

Version 0.4.1

Description Visualizations for SHAP (SHapley Additive exPlanations), such
as waterfall plots, force plots, various types of importance plots,
and dependence 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', 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 7.2.1

Imports ggfittext (>= 0.8.0), gggenes, ggplot2 (>= 3.0.0), ggrepel,
grid, rlang (>= 0.3.0), stats, utils, xgboost

Enhances h2o, lightgbm

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL `https://github.com/mayer79/shapviz`

BugReports `https://github.com/mayer79/shapviz/issues`

NeedsCompilation no

Author Michael Mayer [aut, cre]

Maintainer Michael Mayer <`mayermichael79@gmail.com`>

Repository CRAN

Date/Publication 2023-01-11 18:20:05 UTC

R topics documented:

shapviz-package 2
collapse_shap 3
dim.shapviz 4
extractors 4
format_max 5
is.shapviz 6
potential_interactions 7
print.shapviz 8
shapviz 8
sv_dependence 12
sv_force 13
sv_importance 15
sv_waterfall 17

Index **20**

`shapviz-package` *shapviz: SHAP Visualizations*

Description

Visualizations for SHAP (SHapley Additive exPlanations), such as waterfall plots, force plots, various types of importance plots, and dependence 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', 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.

Author(s)

Maintainer: Michael Mayer <`mayermichael79@gmail.com`>

See Also

Useful links:

- `https://github.com/mayer79/shapviz`
- Report bugs at `https://github.com/mayer79/shapviz/issues`

`collapse_shap` *Collapse SHAP values*

Description

Function used to collapse groups of columns in a SHAP matrix by rowwise summation. A typical application is when the matrix of SHAP values is generated by a model with one or multiple one-hot encoded variables and the explanations should be done using the original variables.

Usage

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

Arguments

`S`	A matrix of SHAP values.
`collapse`	A named list of character vectors. Each vector specifies a group of column names in `S` that should be collapsed to a single column by summation. The name of the new column equals the name of the vector in `collapse`.
`...`	Currently unused.

Value

A matrix with collapsed columns.

See Also

`shapviz`.

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)
```

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 (or the feature dataset) 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))
dim(shapviz(S, X))
```

extractors *Extractor Functions*

Description

Functions to extract SHAP values, feature values, or the baseline from a "shapviz" object.

Usage

```
get_shap_values(object, ...)

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

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

get_feature_values(object, ...)

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

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

get_baseline(object, ...)

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

## Default S3 method:
get_baseline(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, and `get_baseline()` the numeric baseline value of the input.

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` *Number Formatter*

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.shapviz` *Check for shapviz*

Description

Is object of class "shapviz"?

Usage

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

Arguments

`object`	An R object.

Value

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

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)
is.shapviz(shp)
is.shapviz("a")
```

`potential_interactions` *Strongest Interaction*

Description

This function tries to detect the approximately strongest interacting feature with `v`. It works by calculating an average squared correlation between the SHAP values of `v` and each feature across values of `v`. To this purpose, a numeric `v` with more than `n_bins` unique values is binned into that many quantile bins. Currently `n_bins` equals the smaller of n/20 and sqrt(n), where n is the sample size. The average squared correlation is weighted by the number of non-missing feature values in the bin. Note that non-numeric color features are turned to numeric by calling `data.matrix`, which does not necessarily make sense.

Usage

```
potential_interactions(obj, v)
```

Arguments

`obj`	An object of type "shapviz".
`v`	Variable name.

Value

A named vector of average squared correlations, sorted in decreasing order.

See Also

`sv_dependence`

`print.shapviz` *Prints "shapviz" Object*

Description

Prints "shapviz" Object

Usage

```
## S3 method for class 'shapviz'
print(x, n = 2L, ...)
```

Arguments

`x`	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

```
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)
```

`shapviz` *Initialize "shapviz" Object*

Description

This function creates an object of class "shapviz" from one of the following inputs:

- Matrix with SHAP values
- XGBoost model
- LightGBM model
- "explain" object from the package "fastshap"
- H2O model (tree-based regression or binary classification model)

- "shapr" object from the package "shapr"
- The result of calling `treeshap()` from the "treeshap" package
- "kernelshap" object from the "kernelshap" package

The "shapviz" vignette explains how to use each of them. Together with the main input, a data set `X` of feature values is required, which is 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.

Usage

```
shapviz(object, ...)

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

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

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

## 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, baseline = 0, collapse = NULL, ...)

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

## 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. It 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 a group of column names in the SHAP matrix that should be collapsed to a single column by summation. The name of the new column equals the name of the vector in `collapse`.
`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, and kernelshap.

Value

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

- `S`: A numeric matrix of SHAP values.
- `X`: A `data.frame` containing the feature values corresponding to `S`.
- `baseline`: Baseline value, representing the average prediction at the scale of the SHAP values.

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's "explain()" method.
- `shapviz(treeshap)`: Creates a "shapviz" object from treeshap's "treeshap()" method.
- `shapviz(shapr)`: Creates a "shapviz" object from shapr's "explain()" method.
- `shapviz(kernelshap)`: Creates a "shapviz" object from kernelshap's "kernelshap()" method.

- `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_waterfall`, `sv_force`, `collapse_shap`

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))
shapviz(S, X, baseline = 4)

X_pred <- data.matrix(iris[, -1])
dtrain <- xgboost::xgb.DMatrix(X_pred, label = iris[, 1])
fit <- xgboost::xgb.train(data = dtrain, nrounds = 50)

# Will use numeric matrix "X_pred" as feature matrix
x <- shapviz(fit, X_pred = X_pred)
x
sv_dependence(x, "Species")

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

# "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)

# Similarly with LightGBM
if (requireNamespace("lightgbm", quietly = TRUE)) {
  fit <- lightgbm::lgb.train(
    params = list(objective = "regression"),
    data = lightgbm::lgb.Dataset(X_pred, label = iris[, 1]),
    nrounds = 50,
    verbose = -2
  )
  x <- shapviz(fit, X_pred = X_pred)
}

# In multiclass setting, we need to specify which_class (integer starting at 1)
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]) - 1L)
fit <- xgboost::xgb.train(params = params, data = dtrain, nrounds = 50)
x <- shapviz(fit, X_pred = X_pred, which_class = 3)

# 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]))
fit <- xgboost::xgb.train(data = dtrain, nrounds = 50)
x <- shapviz(
  fit,
  X_pred = X_pred,
  X = iris,
  collapse = list(Species = c("Speciessetosa", "Speciesversicolor", "Speciesvirginica"))
)
x
```

`sv_dependence` *SHAP Dependence Plot*

Description

Creates a scatter plot of the SHAP values of a feature against its feature values. A second variable, `color_var`, can be selected to be used on the color axis. In this way, one can get a sense of possible interaction effects. Set `color_var = "auto"` to use a simple heuristic to select the color feature with the strongest apparent interaction. With discrete `v`, horizontal jitter is added by default.

Usage

```
sv_dependence(object, ...)

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

## S3 method for class 'shapviz'
sv_dependence(
  object,
  v,
  color_var = NULL,
  color = "#3b528b",
  viridis_args = getOption("shapviz.viridis_args"),
  jitter_width = NULL,
  ...
)
```

Arguments

`object`	An object of class "shapviz".
`...`	Arguments passed to `geom_jitter()`.
`v`	Column name of feature to be plotted.
`color_var`	Feature name to be used on the color scale to investigate interactions. The default is `NULL` (no color feature). An experimental option is "auto", which selects - by a simple heuristic - a variable with seemingly strongest interaction. Check details for how to change the color scale.

`color`	Color to be used if `color_var = NULL`.
`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 with `options(shapviz.viridis_args = NULL)` or set `viridis_args = NULL`. 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 a factor, logical, or character variable, and no jitter otherwise.

Value

An object of class `ggplot` representing a dependence plot.

Methods (by class)

- `sv_dependence(default)`: Default method.
- `sv_dependence(shapviz)`: SHAP dependence plot for shp object.

See Also

`potential_interactions`

Examples

```
dtrain <- xgboost::xgb.DMatrix(data.matrix(iris[, -1]), label = iris[, 1])
fit <- xgboost::xgb.train(data = dtrain, nrounds = 50)
x <- shapviz(fit, X_pred = dtrain, X = iris[, -1])
sv_dependence(x, "Petal.Length")
sv_dependence(x, "Petal.Length", color_var = "Species")
sv_dependence(x, "Species", color_var = "auto")
```

`sv_force` *SHAP Force Plot*

Description

Creates a force plot of SHAP values of one single observation. The value of f(x) denotes the prediction on the SHAP scale, while E(f(x)) refers to the baseline SHAP value.

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,
  ...
)
```

Arguments

`object`	An object of class "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`	A single row number to plot.
`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 detemines 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))=...).

Value

An object of class "ggplot" representing a force plot.

Methods (by class)

- `sv_force(default)`: Default method.
- `sv_force(shapviz)`: SHAP force plot for object of class "shapviz".

See Also

sv_waterfall

Examples

```
dtrain <- xgboost::xgb.DMatrix(data.matrix(iris[, -1]), label = iris[, 1])
fit <- xgboost::xgb.train(data = dtrain, nrounds = 50)
x <- shapviz(fit, X_pred = dtrain, X = iris[, -1])
sv_force(x)
sv_force(x, row_id = 65, max_display = 3, size = 9, fill_colors = 4:5)
```

---

sv_importance *SHAP Importance Plots*

Description

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

Usage

```
sv_importance(object, ...)

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

## S3 method for class 'shapviz'
sv_importance(
  object,
  kind = c("bar", "beeswarm", "both", "no"),
  max_display = 15L,
  show_other = TRUE,
  fill = "#fca50a",
  bar_width = 2/3,
  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,
  ...
)
```

Arguments

`object`	An object of class "shapviz".
`...`	Arguments passed to `geom_bar()` (if `kind = "bar"`) or to `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`	Maximum number of features (with highest importance) to plot. If there are more, the least important variables are collapsed to an "other" group: their SHAP values are added and their min-max-scaled feature values are added as well (and the resulting vector is min-max-scaled again). Set to `Inf` to show all features. Has no effect if `kind = "no"` or if `show_other = FALSE`.
`show_other`	If the number of features is larger than `max_display`: Should the "other" group be shown (default) or not?
`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 (only used if beeswarm shown).
`bee_adjust`	Relative bandwidth adjustment factor used in estimating the density of the beeswarms (only used if beeswarm shown).
`viridis_args`	List of viridis color scale arguments used to control the coloring of the beeswarm plot, 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_c()`. For example, to switch to a standard viridis scale, you can either change the default with `options(shapviz.viridis_args = NULL)` or set `viridis_args = NULL`.
`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 e.g. `function(x) = prettyNum(x, scientific = TRUE)`.
`number_size`	Text size of the numbers (if `show_numbers = TRUE`).

Value

A "ggplot" object representing an importance plot, or - if `kind = "no"` - a named numeric vector of sorted SHAP feature importances.

Methods (by class)

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

Examples

```
X_train <- data.matrix(iris[, -1])
dtrain <- xgboost::xgb.DMatrix(X_train, label = iris[, 1])
fit <- xgboost::xgb.train(data = dtrain, nrounds = 50)
x <- shapviz(fit, X_pred = X_train)
sv_importance(x)
sv_importance(x, kind = "beeswarm", show_numbers = TRUE)
sv_importance(x, kind = "no")

X <- data.frame(matrix(rnorm(1000), ncol = 20))
S <- as.matrix(X)
x2 <- shapviz(S, X)
sv_importance(x2)
sv_importance(x2, max_display = 5)
sv_importance(x2, max_display = 5, show_other = FALSE)
```

`sv_waterfall` *SHAP Waterfall Plot*

Description

Creates a waterfall plot of SHAP values of one single observation. The value of f(x) denotes the prediction on the SHAP scale, while E(f(x)) refers to the baseline SHAP value. The plot has to be read from bottom to top.

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,
  ...
)
```

Arguments

`object`	An object of class "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`	A single row number to plot.
`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. The default is ten in order to not overload the plot. 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 detemines 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))=...).

Value

An object of class "ggplot" representing a waterfall plot.

Methods (by class)

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

See Also

sv_force

Examples

```
dtrain <- xgboost::xgb.DMatrix(data.matrix(iris[, -1]), label = iris[, 1])
fit <- xgboost::xgb.train(data = dtrain, nrounds = 50)
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)

X <- as.data.frame(matrix(1:100, nrow = 10))
S <- as.matrix(X)
shp <- shapviz(S, X)
sv_waterfall(shp)
```

Index

_PACKAGE (shapviz-package), 2

collapse_shap, 3, *11*

dim.shapviz, 4

extractors, 4

format_max, 5

get_baseline (extractors), 4
get_feature_values (extractors), 4
get_shap_values (extractors), 4

is.shapviz, 6

potential_interactions, 7, *13*
print.shapviz, 8

shapviz, *3*, *4*, *8*, 8
shapviz-package, 2
sv_dependence, *7*, *11*, 12
sv_force, *11*, 13, *19*
sv_importance, *11*, 15
sv_waterfall, *11*, *15*, 17