

Package: fastshap (via r-universe)

September 22, 2026

Type Package

Title Fast Approximate Shapley Values

Version 0.3.0

Description Computes fast (relative to other implementations) approximate Shapley values for any supervised learning model. Shapley values help to explain the predictions from any black box model using ideas from game theory; see Strumbelj and Kononenko (2014) <[doi:10.1007/s10115-013-0679-x](https://doi.org/10.1007/s10115-013-0679-x)> for details.

License GPL (>= 2)

URL <https://github.com/bgreenwell/fastshap>,
<https://bgreenwell.github.io/fastshap/>,
<https://bgreenwell.r-universe.dev/fastshap>

BugReports <https://github.com/bgreenwell/fastshap/issues>

Imports Rcpp (>= 1.0.1), stats, utils

Enhances lightgbm, xgboost

Suggests covr, doParallel, earth, foreach, knitr, ranger, rmarkdown, shapviz (>= 0.8.0), tinytest (>= 1.4.1)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.3

Encoding UTF-8

Depends R (>= 3.6.0)

LazyData true

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Repository <https://bgreenwell.r-universe.dev>

Date/Publication 2026-07-06 03:27:16 UTC

RemoteUrl <https://github.com/bgreenwell/fastshap>

RemoteRef main

RemoteSha fe2193ce5b17ea64fe60ff88e919417f434c4f91

Contents

explain	2
explain_effects	7
gen_friedman	9
titanic	10
titanic_mice	11
Index	12

explain	<i>Fast approximate Shapley values</i>
---------	--

Description

Compute fast (approximate) Shapley values for a set of features using the Monte Carlo algorithm described in Strumbelj and Igor (2014). An efficient algorithm for tree-based models, commonly referred to as Tree SHAP, is also supported for **lightgbm** and **xgboost** models; see Lundberg et. al. (2020) for details.

Usage

```
explain(object, ...)

## Default S3 method:
explain(
  object,
  feature_names = NULL,
  X = NULL,
  nsim = 1,
  pred_wrapper = NULL,
  newdata = NULL,
  adjust = FALSE,
  baseline = NULL,
  shap_only = TRUE,
  parallel = FALSE,
  raw = FALSE,
  seed = NULL,
  exact = FALSE,
  batch_size = NULL,
  method = c("strumbelj", "permutation"),
  ...
)

## S3 method for class 'lm'
explain(
  object,
  feature_names = NULL,
```

```
X,
  nsim = 1,
  pred_wrapper,
  newdata = NULL,
  adjust = FALSE,
  exact = FALSE,
  baseline = NULL,
  shap_only = TRUE,
  parallel = FALSE,
  ...
)

## S3 method for class 'xgb.Booster'
explain(
  object,
  feature_names = NULL,
  X = NULL,
  nsim = 1,
  pred_wrapper,
  newdata = NULL,
  adjust = FALSE,
  exact = FALSE,
  baseline = NULL,
  shap_only = TRUE,
  parallel = FALSE,
  ...
)

## S3 method for class 'lgb.Booster'
explain(
  object,
  feature_names = NULL,
  X = NULL,
  nsim = 1,
  pred_wrapper,
  newdata = NULL,
  adjust = FALSE,
  exact = FALSE,
  baseline = NULL,
  shap_only = TRUE,
  parallel = FALSE,
  ...
)
```

Arguments

object	A fitted model object (e.g., a <code>ranger::ranger()</code> , <code>xgboost::xgboost()</code> , or <code>earth::earth()</code> object, to name a few).
--------	---

...	Additional optional arguments to be passed on to <code>foreach::foreach()</code> whenever <code>parallel = TRUE</code> (e.g., <code>.packages</code> for loading packages on the workers); ignored otherwise.
<code>feature_names</code>	Character string giving the names of the predictor variables (i.e., features) of interest. If <code>NULL</code> (default) they will be taken from the column names of <code>X</code> .
<code>X</code>	A matrix-like R object (e.g., a data frame or matrix) containing ONLY the feature columns from the training data (or suitable background data set). NOTE: This argument is required whenever <code>exact = FALSE</code> .
<code>nsim</code>	The number of Monte Carlo repetitions to use for estimating each Shapley value (only used when <code>exact = FALSE</code>). Default is 1. NOTE: To obtain the most accurate results, <code>nsim</code> should be set as large as feasibly possible.
<code>pred_wrapper</code>	Prediction function that requires two arguments, <code>object</code> and <code>newdata</code> . NOTE: This argument is required whenever <code>exact = FALSE</code> . The output of this function should be determined according to: Regression A numeric vector of predicted outcomes. Binary classification A vector of predicted class probabilities for the reference class. Multiclass classification A vector of predicted class probabilities for the reference class.
<code>newdata</code>	A matrix-like R object (e.g., a data frame or matrix) containing ONLY the feature columns for the observation(s) of interest; that is, the observation(s) you want to compute explanations for. Default is <code>NULL</code> which will produce approximate Shapley values for all the rows in <code>X</code> (i.e., the training data).
<code>adjust</code>	Logical indicating whether or not to adjust the sum of the estimated Shapley values to satisfy the <i>local accuracy</i> property; that is, to equal the difference between the model's prediction for that sample and the average prediction over all the training data (i.e., <code>X</code>). Default is <code>FALSE</code> and setting to <code>TRUE</code> requires <code>nsim > 1</code> .
<code>baseline</code>	Numeric baseline to use when adjusting the computed Shapley values to achieve <i>local accuracy</i> . Adjusted Shapley values for a single prediction (<code>fx</code>) will sum to the difference <code>fx - baseline</code> . Defaults to <code>NULL</code> , which corresponds to the average predictions computed from <code>X</code> , and zero otherwise (i.e., no additional predictions will be computed and the baseline attribute of the output will be set to zero).
<code>shap_only</code>	Logical indicating whether or not to include additional output useful for plotting (i.e., <code>newdata</code> and the baseline value.). This is convenient, for example, when using <code>shapviz::shapviz()</code> for plotting. Default is <code>TRUE</code> .
<code>parallel</code>	Logical indicating whether or not to compute the approximate Shapley values in parallel across features; default is <code>FALSE</code> . NOTE: setting <code>parallel = TRUE</code> requires the <code>foreach</code> package (in <code>Suggests</code>) and an appropriate (i.e., system-specific) <i>parallel backend</i> (e.g., via the <code>doParallel</code> package). For reproducible parallel results, use the <code>doRNG</code> package to register the backend. Note that the internal batching of predictions (see <code>batch_size</code>) means models whose <code>predict()</code> methods are already multithreaded (e.g., <code>ranger</code> or <code>xgboost</code>) often gain little from <code>parallel = TRUE</code> . The <code>fastshap</code> namespace is always added to

	<code>.packages</code> automatically (socket-based clusters, the default on Windows, start with no packages attached), so any <i>additional</i> packages needed by <code>pred_wrapper</code> should be supplied via <code>.packages</code> as usual. If <code>fastshap</code> (or a dependency) is installed somewhere other than the default library path (e.g., a development checkout, <code>renv/pak</code> project library, or during <code>R CMD check</code>, which installs the package under test into an ephemeral library), fresh workers won't see it either; propagate the current session's library path to the cluster before registering it, e.g. <code>parallel::clusterCall(cl, function(lp) .libPaths(lp), .libPaths())</code>.
<code>raw</code>	Logical indicating whether or not to return the raw per-simulation Shapley values from each Monte Carlo replication. If <code>TRUE</code>, a 3-D array of dimensions $n \times p \times \text{nsim}$ is returned, where n is the number of observations, p is the number of features, and nsim is the number of Monte Carlo replications; for example, <code>apply(result, 1:2, sd)</code> computes standard errors for each (observation, feature) pair. Only supported when <code>adjust = FALSE</code>. Default is <code>FALSE</code>.
<code>seed</code>	Integer specifying a random seed for reproducibility; passed to <code>base::set.seed()</code>. Default is <code>NULL</code> (no seed). NOTE: the Monte Carlo loop was restructured (vectorized) in version 0.2.0, so seeded results differ from those produced by <code>fastshap</code> ($\leq 0.1.5$).
<code>exact</code>	Logical indicating whether to compute exact Shapley values. Currently only supported for <code>stats::lm()</code>, <code>xgboost::xgboost()</code>, and <code>lightgbm::lightgbm()</code> objects (binary/regression only — multiclass is not yet supported). Passing <code>exact = TRUE</code> for any other model type issues a warning and falls back to the Monte Carlo approximation. Note that <code>exact = TRUE</code> for <code>stats::lm()</code> returns explanations for each of the <code>stats::terms()</code> in the model. Default is <code>FALSE</code>.
<code>batch_size</code>	Optional positive integer giving the maximum number of rows to pass to <code>pred_wrapper()</code> per call. By default (<code>NULL</code>), all nsim Monte Carlo replications for a feature are stacked and evaluated in a single pair of prediction calls, which is fastest but requires $2 * \text{nrow}(\text{newdata}) * \text{nsim} * \text{ncol}(X)$ values of working memory per feature; set <code>batch_size</code> to bound the size of each prediction call instead. All of the randomness is drawn up front, so for a fixed seed the results are identical regardless of <code>batch_size</code>.
<code>method</code>	Character string specifying the Monte Carlo estimator to use (only relevant when <code>exact = FALSE</code>). <code>"strumbelj"</code> (the default) is the Strumbelj and Kononenko (2014) estimator described above: independent random coalitions per feature, $2 * \text{nsim}$ predictions per feature ($2 * \text{nsim} * p$ total for p features), and supports <code>feature_names</code> subsetting (requesting fewer features costs proportionally less) and <code>parallel = TRUE</code>. <code>"permutation"</code> walks a single random permutation through <i>all</i> features per replication, flipping one feature at a time from background to foreground and reading off each feature's contribution as the consecutive prediction difference; this costs about $\text{nsim} * p$ predictions <i>for all features combined</i> (roughly half of <code>"strumbelj"</code>'s cost for computing every feature), and each replication's contributions sum <i>exactly</i> to $f(x) - f(\text{background})$ (verifiable via <code>raw = TRUE</code>), so <code>adjust = TRUE</code> is inapplicable (and an error) for this method. Because the telescoping property requires walking every feature regardless of what is requested, <code>feature_names</code> only subsets the <i>output</i> under <code>"permutation"</code> — it does not reduce the amount of work performed, unlike <code>"strumbelj"</code>. <code>parallel = TRUE</code> is not yet supported for <code>"permutation"</code>.

Value

If `shap_only = TRUE` (the default), a matrix is returned with one column for each feature specified in `feature_names` (if `feature_names = NULL`, the default, there will be one column for each feature in `X`) and one row for each observation in `newdata` (if `newdata = NULL`, the default, there will be one row for each observation in `X`). Additionally, the returned matrix will have an attribute called "baseline" containing the baseline value. If `shap_only = FALSE`, then a list is returned with three components:

- `shapley_values` - a matrix of Shapley values (as described above);
- `feature_values` - the corresponding feature values (for plotting with `shapviz::shapviz()`);
- `baseline` - the corresponding baseline value (for plotting with `shapviz::shapviz()`).

Note

Setting `exact = TRUE` with a linear model (i.e., an `stats::lm()` or `stats::glm()` object) assumes that the input features are independent. Also, setting `adjust = TRUE` is experimental and we follow the same approach as in [shap](#).

References

Strumbelj, E., and Igor K. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3), 647-665.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, Su-In (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1), 2522–5839.

See Also

You can find more examples (with larger and more realistic data sets) on the **fastshap** GitHub repository: <https://github.com/bgreenwell/fastshap>.

Examples

```
#
# A projection pursuit regression (PPR) example
#

# Load the sample data; see ?datasets::mtcars for details
data(mtcars)

# Fit a projection pursuit regression model
fit <- ppr(mpg ~ ., data = mtcars, nterms = 5)

# Prediction wrapper
pfun <- function(object, newdata) { # needs to return a numeric vector
  predict(object, newdata = newdata)
}

# Compute approximate Shapley values using 10 Monte Carlo simulations
set.seed(101) # for reproducibility
```

```
shap <- explain(fit, X = subset(mtcars, select = -mpg), nsim = 10,
               pred_wrapper = pfun)
head(shap)
```

explain_effects	Main, total, and interaction effect decomposition
-----------------	---

Description

For each feature, decompose its contribution to a prediction into a *main effect* (the feature's effect in isolation), a *total effect* (the feature's effect together with every interaction it participates in), and the gap between them (an *interaction* diagnostic). Unlike `explain()`, which estimates Shapley values by averaging over *randomly sampled* coalitions, `explain_effects()` uses only the two coalitions with the highest Shapley weight — the empty coalition and the full coalition — so by default (`nsim = NULL`) it is deterministic given the background data, with no Monte Carlo simulation at all.

Usage

```
explain_effects(
  object,
  feature_names = NULL,
  X = NULL,
  pred_wrapper = NULL,
  newdata = NULL,
  nsim = NULL,
  batch_size = NULL,
  seed = NULL,
  ...
)

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

Arguments

object	A fitted model object (e.g., a <code>ranger::ranger()</code> , <code>xgboost::xgboost()</code> , or <code>earth::earth()</code> object, to name a few).
feature_names	Character string giving the names of the predictor variables (i.e., features) of interest. If <code>NULL</code> (default) they will be taken from the column names of <code>X</code> .
X	A matrix-like R object (e.g., a data frame or matrix) containing ONLY the feature columns from the training data (or suitable background data set). NOTE: This argument is required whenever <code>exact = FALSE</code> .
pred_wrapper	Prediction function that requires two arguments, <code>object</code> and <code>newdata</code> . NOTE: This argument is required whenever <code>exact = FALSE</code> . The output of this function should be determined according to: Regression A numeric vector of predicted outcomes.

	Binary classification A vector of predicted class probabilities for the reference class.
	Multiclass classification A vector of predicted class probabilities for the reference class.
newdata	A matrix-like R object (e.g., a data frame or matrix) containing ONLY the feature columns for the observation(s) of interest; that is, the observation(s) you want to compute explanations for. Default is NULL which will produce approximate Shapley values for all the rows in X (i.e., the training data).
nsim	Optional positive integer giving the number of background rows to sample per observation. Default (NULL) averages over <i>every</i> row of X (deterministic, exact given the background data, but requires $2 * \text{nrow}(\text{newdata}) * \text{length}(\text{feature_names}) * \text{nrow}(X)$ predictions); set nsim to average over a random sample of that many background rows instead (requires seed for reproducibility).
batch_size	Optional positive integer giving the maximum number of rows to pass to <code>pred_wrapper()</code> per call. By default (NULL), all nsim Monte Carlo replications for a feature are stacked and evaluated in a single pair of prediction calls, which is fastest but requires $2 * \text{nrow}(\text{newdata}) * \text{nsim} * \text{ncol}(X)$ values of working memory per feature; set batch_size to bound the size of each prediction call instead. All of the randomness is drawn up front, so for a fixed seed the results are identical regardless of batch_size.
seed	Integer specifying a random seed for reproducibility; passed to <code>base::set.seed()</code> . Default is NULL (no seed). NOTE: the Monte Carlo loop was restructured (vectorized) in version 0.2.0, so seeded results differ from those produced by fastshap ($\leq 0.1.5$).
...	Additional optional arguments to be passed on to <code>foreach::foreach()</code> whenever <code>parallel = TRUE</code> (e.g., <code>.packages</code> for loading packages on the workers); ignored otherwise.
x	An object of class "explain_effects".

Details

For feature j and observation x , averaging over background draws w :

main $E_w[f(x_j, w_{-j})] - E_w[f(w)]$. Since no other feature of x is present, this is free of interaction effects — it is exactly the centered *partial dependence* of feature j evaluated at x_j .

total $f(x) - E_w[f(w_j, x_{-j})]$. This captures the main effect of feature j together with every interaction it participates in, in the context of x .

interaction total - main. Zero exactly when feature j enters the model additively (no interactions involving j) at x ; for a two-way interaction the gap splits equally between the two participating features.

shapley_values (main + total) / 2. This equals the exact (background-marginal) Shapley value for any model with at most pairwise interactions; for higher-order interactions it is a cheap but biased approximation.

As with `explain()`, this uses marginal (interventional) sampling of the background, so the usual caveat applies: with correlated features, the constructed hybrid observations can be unrealistic combinations of feature values.

Value

An object of class "explain_effects": a list with components main, total, and interaction (each an $nrow(newdata) \times length(feature_names)$ matrix), shapley_values (same shape), and baseline (the average prediction over X).

References

Strumbelj, E., and Igor K. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3), 647-665.

Examples

```
data(mtcars)
fit <- ppr(mpg ~ ., data = mtcars, nterms = 5)
pfun <- function(object, newdata) predict(object, newdata = newdata)
eff <- explain_effects(fit, X = subset(mtcars, select = -mpg),
                      pred_wrapper = pfun)

eff
```

gen_friedman	<i>Friedman benchmark data</i>
--------------	--------------------------------

Description

Simulate data from the Friedman 1 benchmark problem. These data were originally described in Friedman (1991) and Breiman (1996). For details, see [sklearn.datasets.make_friedman1](#).

Usage

```
gen_friedman(n_samples = 100, n_features = 10, sigma = 0.1, seed = NULL)
```

Arguments

n_samples	Integer specifying the number of samples (i.e., rows) to generate. Default is 100.
n_features	Integer specifying the number of features to generate. Default is 10.
sigma	Numeric specifying the standard deviation of the noise.
seed	Integer specifying the random seed. If NULL (the default) the results will be different each time the function is run.

Note

This function is mostly used for internal testing.

References

Breiman, Leo (1996) Bagging predictors. *Machine Learning* 24, pages 123-140.
 Friedman, Jerome H. (1991) Multivariate adaptive regression splines. *The Annals of Statistics* 19 (1), pages 1-67.

Examples

```
gen_friedman()
```

titanic	<i>Survival of Titanic passengers</i>
---------	---------------------------------------

Description

A data set containing the survival outcome, passenger class, age, sex, and the number of family members for a large number of passengers aboard the ill-fated Titanic.

Usage

```
titanic
```

Format

A data frame with 1309 observations on the following 6 variables:

survived binary with levels "yes" for survived and "no" otherwise;

pclass integer giving the corresponding passenger (i.e., ticket) class with values 1–3;

age the age in years of the corresponding passenger (with 263 missing values);

sex factor giving the sex of each passenger with levels "male" and "female";

sibsp integer giving the number of siblings/spouses aboard for each passenger (ranges from 0–8);

parch integer giving the number of parents/children aboard for each passenger (ranges from 0–9).

Note

As mentioned in the column description, age contains 263 NAs (or missing values). For a complete version (or versions) of the data set, see [titanic_mice](#).

Source

<https://hbiostat.org/data/>.

titanic_mice	<i>Survival of Titanic passengers</i>
--------------	---------------------------------------

Description

The [titanic](#) data set contains 263 missing values (i.e., NA's) in the age column. This version of the data contains imputed values for the age column using *multivariate imputation by chained equations* via the [mice](#) package. Consequently, this is a list containing 11 imputed versions of the observations contained in the [titanic](#) data frame; each completed data sets has the same dimension and column structure as [titanic](#).

Usage

```
titanic_mice
```

Format

An object of class `mild` (inherits from `list`) of length 21.

Source

Greenwell, Brandon M. (2022). *Tree-Based Methods for Statistical Learning in R*. CRC Press.

Index

- * **datasets**
 - titanic, [10](#)
 - titanic_mice, [11](#)
- base::set.seed(), [5](#), [8](#)
- earth::earth(), [3](#), [7](#)
- explain, [2](#)
- explain(), [7](#), [8](#)
- explain_effects, [7](#)
- foreach::foreach(), [4](#), [8](#)
- gen_friedman, [9](#)
- lightgbm::lightgbm(), [5](#)
- print.explain_effects
 (explain_effects), [7](#)
- ranger::ranger(), [3](#), [7](#)
- shapviz::shapviz(), [4](#), [6](#)
- stats::glm(), [6](#)
- stats::lm(), [5](#), [6](#)
- stats::terms(), [5](#)
- titanic, [10](#), [11](#)
- titanic_mice, [10](#), [11](#)
- xgboost::xgboost(), [3](#), [5](#), [7](#)