

Package: figsr (via r-universe)

October 17, 2026

Title Fast Interpretable Greedy-Tree Sums for Tree Ensembles

Version 0.2.0.9000

Description Flexible, interpretable machine learning algorithm for additive tree sums (FIGS). Fits a sum of shallow classification and regression trees (CART) by greedily minimizing residual impurity, growing a new tree or deepening an existing one at each step, whichever reduces the residuals most. Supports regression and two-class classification, variable importance, bootstrap ensembling and seamless integration with 'parsnip' and 'tidymodels' workflows. The method is described in Tan et al. (2023) <[doi:10.1073/pnas.2310151122](https://doi.org/10.1073/pnas.2310151122)>.

License MIT + file LICENSE

URL <https://github.com/bonijoao/figsr>

BugReports <https://github.com/bonijoao/figsr/issues>

Depends R (>= 4.1)

Imports dials, graphics, parsnip, rlang, stats, tibble

Suggests knitr, recipes, rmarkdown, testthat (>= 3.0.0), workflows

VignetteBuilder knitr, rmarkdown

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-10-17 02:35:00 UTC

RemoteUrl <https://github.com/bonijoao/figsr>

RemoteRef HEAD

RemoteSha 1a263da5b87f3937f9cfa385dc9809fe49caf997

Contents

bagging_figs	2
figs	3
figs_tree	5
figsr_importance	6
fit_figs	7
max_splits	8
max_trees	9
pima_diabetes	9
plot.figsr_fit	10
predict.bagging_figs_fit	12
predict.figsr_fit	12
predict_figs	13
print.figsr_fit	14
summary.figsr_fit	14
update.figs_tree	15
Index	16

bagging_figs	<i>Bagging-FIGS: Ensemble of FIGS Models</i>
--------------	--

Description

bagging_figs() fits an ensemble of FIGS models using bootstrap resampling to increase stability and predictive performance on noisy datasets.

Usage

```
bagging_figs(  
  formula,  
  data,  
  n_estimators = 10,  
  max_splits = 6,  
  min_n = 5,  
  mode = "regression",  
  na_method = c("omit", "mia"),  
  ...  
)
```

Arguments

formula	A formula.
data	A data frame.
n_estimators	Integer. Number of bootstrap FIGS models to train. Default is 10.
max_splits	Integer. Maximum splits per individual FIGS model. Default is 6.

<code>min_n</code>	Integer. Minimum node size. Default is 5.
<code>mode</code>	Character. "regression" or "classification"; classification supports two-class outcomes only.
<code>na_method</code>	Character. "omit" or "mia", passed to every member fit; see <code>figs()</code> .
<code>...</code>	Additional arguments passed to <code>figs()</code> .

Value

An object of class `bagging_figs_fit`.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(50), x2 = rnorm(50), y = rnorm(50))
bag_fit <- bagging_figs(y ~ x1 + x2, data = df, n_estimators = 3)
```

<code>figs</code>	<i>Fit Fast Interpretable Greedy-Tree Sums (FIGS)</i>
-------------------	---

Description

`figs()` fits a Fast Interpretable Greedy-Tree Sums model for regression or binary classification tasks.

Usage

```
figs(
  formula,
  data,
  max_splits = 10,
  max_trees = NULL,
  min_n = 5,
  mode = "regression",
  subset = NULL,
  na.action = stats::na.omit,
  na_method = c("omit", "mia"),
  ...
)
```

Arguments

<code>formula</code>	A formula specifying outcome and predictor variables.
<code>data</code>	A data frame containing training data.
<code>max_splits</code>	Integer. Maximum total number of splits across all trees in the sum. Default is 10.

<code>max_trees</code>	Integer. Maximum number of trees allowed in the sum. Default is NULL (unconstrained up to <code>max_splits</code>).
<code>min_n</code>	Integer. Minimum number of observations required in a node to split. Default is 5.
<code>mode</code>	Character. Either "regression" or "classification". Only two-class outcomes are supported in classification mode. Default is "regression".
<code>subset</code>	An optional vector selecting the rows of data to fit on, passed to <code>stats::model.frame()</code> .
<code>na.action</code>	A function describing what to do with missing values, passed to <code>stats::model.frame()</code> . Defaults to <code>stats::na.omit()</code> . Cannot be combined with <code>na_method = "mia"</code> .
<code>na_method</code>	Character. How to treat missing predictor values: "omit" (the default) drops incomplete rows, "mia" learns a direction for them at every split and can also split on missingness itself. See Details. Cannot be combined with <code>na.action</code> .
<code>...</code>	Additional arguments, currently ignored. Case weights are not supported and passing weights raises an error.

Details

Trees are grown one split at a time. At each step every leaf of every existing tree, plus the root of a possible new tree, competes for the split that most reduces the residual sum of squares; the single best candidate is taken and the residuals are recomputed against the whole sum. Growth stops at `max_splits` splits or when no candidate improves the fit.

Classification is limited to two classes. A factor outcome with more than two levels raises an error. The binary case is fitted on the 0/1 encoding of the outcome using the same sum-of-squares criterion, so the sum of the leaf values already estimates the probability of the second level; it is clamped to the unit interval at prediction time rather than passed through a link function.

Predictors are used exactly as supplied: numeric features are split at midpoints between sorted unique values, capped at 30 sample quantiles, and factor features are split by enumerating subsets of their levels, which is skipped above 10 levels. A factor level that was not seen in training raises an error at prediction time, as it does for the other model-frame based fitting functions in `stats`.

Missing predictor values are governed by `na_method`. With "omit", the default, rows with a missing predictor are dropped at fit time by `stats::model.frame()` and raise an error at prediction time. With "mia" (missingness incorporated in attributes, Twala et al. 2008) every row is kept: for a predictor with missing values the split search also tries sending them to the left or to the right of each cutpoint, and splitting on whether the value is missing at all, and keeps whichever reduces the residual sum of squares most. The learned direction is used at prediction time. A missing value in a predictor that was complete in training raises an error, because no direction was learned; impute such values first, for example with `recipes::step_impute_knn()`. Even under "mia", a branch of the tree that no missing value reached during training has not learned a direction, so a missing value reaching that same branch at prediction time still raises an error. Rows with a missing outcome are dropped with a warning under `na_method = "mia"` only; under "omit" they are dropped silently by `stats::na.omit()`.

Value

An object of class `figsr_fit` containing fitted tree structures, predictions, and metadata.

References

Twala, B. E. T. H., Jones, M. C. and Hand, D. J. (2008). Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7), 950-956.

Examples

```
set.seed(123)
df <- data.frame(
  x1 = rnorm(100),
  x2 = rnorm(100),
  y = rnorm(100)
)
fit <- figs(y ~ x1 + x2, data = df, max_splits = 4)
print(fit)
```

figs_tree

FIGS Model Specification for Parsnip

Description

`figs_tree()` defines a Fast Interpretable Greedy-Tree Sums model for use with the `parsnip` and `tidymodels` ecosystem.

Usage

```
figs_tree(
  mode = "regression",
  engine = "figsr",
  max_splits = NULL,
  max_trees = NULL,
  min_n = NULL
)
```

Arguments

<code>mode</code>	A single character string for the prediction outcome mode: "regression" or "classification". Classification supports two-class outcomes only; a factor with more than two levels raises an error at fit time.
<code>engine</code>	A single character string for the computational engine. Only "figsr" is available.
<code>max_splits</code>	An integer for the maximum total splits across all trees. NULL (the default) leaves the engine default of 10 in place.
<code>max_trees</code>	An integer for the maximum number of trees. NULL (the default) leaves the number of trees unconstrained.
<code>min_n</code>	An integer for the minimum number of data points in a node to split. NULL (the default) leaves the engine default of 5 in place.

Value

A parsnip model specification object.

Examples

```
library(parsnip)
spec <- figs_tree(max_splits = 8) |>
  set_engine("figsr") |>
  set_mode("regression")
spec
```

figsr_importance	<i>Calculate Variable Importance for a figsr_fit Model</i>
------------------	--

Description

`figsr_importance()` computes the total reduction in residual impurity (Sum of Squares gain) attributable to each predictor feature across all trees in the fitted FIGS model.

Usage

```
figsr_importance(object, relative = TRUE)
```

Arguments

<code>object</code>	A fitted <code>figsr_fit</code> model object.
<code>relative</code>	Logical. If TRUE (default), the importance column is rescaled to percentages of the total gain; otherwise it repeats the raw gain.

Details

Each split node stores the reduction in residual sum of squares it achieved at the moment it was chosen. Importance for a feature is the sum of those gains over every split made on that feature, across all trees in the sum. Features never selected receive a gain of zero.

When the model was fitted with `na_method = "mia"`, a split on whether a predictor is missing is reported on its own row, named `missing(<predictor>)`, because such a split never consults the predictor's value. A split that merely sends missing values to one side is credited to the predictor.

Value

A tibble with one row per predictor and columns `feature`, `gain` (raw sum-of-squares reduction) and `importance`, sorted by decreasing importance.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(50), x2 = rnorm(50))
df$y <- 2 * (df$x1 > 0) + rnorm(50, sd = 0.2)
fit <- figs(y ~ x1 + x2, data = df, max_splits = 3)
figsr_importance(fit)
```

fit_figs

Fitting Bridge for the Parsnip Interface

Description

`fit_figs()` is the function `parsnip` calls when fitting a `figs_tree()` specification with the "figsr" engine. It accepts either the formula interface or the x/y interface and forwards to `figs()`. Users normally call `figs()` or `fit()` instead.

Usage

```
fit_figs(
  formula = NULL,
  data = NULL,
  x = NULL,
  y = NULL,
  max_splits = 10,
  max_trees = NULL,
  min_n = 5,
  mode = "regression",
  na_method = c("omit", "mia"),
  ...
)
```

Arguments

<code>formula</code>	A formula specifying outcome and predictors.
<code>data</code>	A data frame containing the training data.
<code>x</code>	A data frame or matrix of predictors.
<code>y</code>	An outcome vector.
<code>max_splits</code>	Integer. Maximum total number of splits across all trees.
<code>max_trees</code>	Integer. Maximum number of trees in the sum.
<code>min_n</code>	Integer. Minimum number of observations in a node to split.
<code>mode</code>	Character. Either "regression" or "classification"; classification supports two-class outcomes only.
<code>na_method</code>	Character. "omit" or "mia"; see <code>figs()</code> . Set it as an engine argument: <code>set_engine("figsr", na_method = "mia")</code> .
<code>...</code>	Additional arguments passed to <code>figs()</code> .

Value

An object of class `figsr_fit`.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(60), x2 = rnorm(60))
df$y <- 2 * (df$x1 > 0) + rnorm(60, sd = 0.2)
fit_figs(x = df[, c("x1", "x2")], y = df$y, max_splits = 3)
```

max_splits

Dials Parameter for Maximum Splits

Description

Dials Parameter for Maximum Splits

Usage

```
max_splits(range = c(2L, 20L), trans = NULL)
```

Arguments

range	A numeric vector of length 2 for lower and upper limits.
trans	A trans object.

Value

A dials parameter object.

Examples

```
max_splits()
max_splits(range = c(2L, 40L))
```

max_trees	<i>Dials Parameter for Maximum Trees</i>
-----------	--

Description

Dials Parameter for Maximum Trees

Usage

```
max_trees(range = c(1L, 10L), trans = NULL)
```

Arguments

range	A numeric vector of length 2 for lower and upper limits.
trans	A trans object.

Value

A dials parameter object.

Examples

```
max_trees()  
max_trees(range = c(1L, 5L))
```

pima_diabetes	<i>Pima Indians Diabetes Database</i>
---------------	---------------------------------------

Description

Diagnostic measurements for 768 women of Pima Indian heritage, aged at least 21, together with whether diabetes was diagnosed within five years. Five columns carry physiologically impossible zeros that are, in fact, missing measurements; they are recoded to NA in this copy of the data, as documented under Details.

Usage

```
pima_diabetes
```

Format

A data frame with 768 rows and 9 columns:

Pregnancies Number of times pregnant.

Glucose Plasma glucose concentration at 2 hours in an oral glucose tolerance test.

BloodPressure Diastolic blood pressure (mm Hg).

SkinThickness Triceps skinfold thickness (mm).

Insulin 2-hour serum insulin (mu U/mL).

BMI Body mass index, weight in kg divided by height in m, squared.

DiabetesPedigreeFunction A score of diabetes likelihood based on family history.

Age Age in years.

Outcome Factor with levels "no" and "yes": whether diabetes was diagnosed within five years.

Details

Glucose, BloodPressure, SkinThickness, Insulin and BMI cannot be zero in a living patient; a 0 recorded for any of them is a missing measurement encoded as if it were data, a well-documented quirk of this data set. Those zeros are recoded to NA here, the standard first preprocessing step for it (see `data-raw/pima_diabetes.R` in the source package for the exact recoding). Insulin is missing in 48.7% of records and SkinThickness in 29.6%; only 51% of rows are complete on all five. Whether a measurement is missing is not unrelated to the outcome; see `vignette("missing-values", package = "figsr")` for a worked example using `na_method = "mia"`.

Source

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the Symposium on Computer Applications and Medical Care*, 261-265. Distributed by the UCI Machine Learning Repository and, from there, on Kaggle as <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>.

Examples

```
data(pima_diabetes)
summary(pima_diabetes)
```

plot.figsr_fit

Plot Method for figsr_fit Models

Description

`plot.figsr_fit()` visualizes individual decision trees comprising the Fast Interpretable Greedy-Tree Sums (FIGS) model using 100% Base R graphics (zero external package dependencies).

Usage

```
## S3 method for class 'figsr_fit'
plot(
  x,
  tree_idx = NULL,
  style = c("scientific", "modern", "classic"),
  main = NULL,
  tree_names = NULL,
  ...
)
```

Arguments

<code>x</code>	A fitted <code>figsr_fit</code> model object.
<code>tree_idx</code>	Optional integer vector specifying indices of trees to plot. Default is <code>NULL</code> (plots all trees).
<code>style</code>	Character string specifying the visual style: "scientific" (default, scientific/editorial notation), "modern" (rounded pastel boxes), or "classic" (high-contrast minimalist statistical style).
<code>main</code>	Optional character string. An overall title drawn once above every tree panel. Default is <code>NULL</code> (no overall title, the previous behavior).
<code>tree_names</code>	Optional character vector overriding the per-tree panel titles ("Subtree f_1(x)", "Tree 1", ...). Either a single string, applied to every plotted tree, or a vector with one entry per tree in <code>x\$trees</code> – indexed by each tree's position in the fit, not by plotting order, so it stays correct when <code>tree_idx</code> only plots some of them. <code>NA</code> in a given position falls back to the default title for that tree only. Default is <code>NULL</code> (the previous, automatic titles).
<code>...</code>	Additional arguments, currently ignored.

Value

Invisible `NULL`.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(50), x2 = rnorm(50), y = rnorm(50))
fit <- figs(y ~ x1 + x2, data = df, max_splits = 3)
plot(fit)
plot(fit, style = "scientific")
plot(fit, tree_idx = 1)
plot(fit, main = "FIGS tree sum", tree_names = "one tree of the sum")
```

```
predict.bagging_figs_fit
```

Predict Method for bagging_figs_fit Models

Description

Predict Method for bagging_figs_fit Models

Usage

```
## S3 method for class 'bagging_figs_fit'
predict(object, new_data, type = NULL, ...)
```

Arguments

object	A fitted bagging_figs_fit object.
new_data	A data frame of predictor values.
type	Character. For classification, either "class" (the default) or "prob"; ignored for regression, which always returns .pred.
...	Additional arguments.

Value

A tibble containing averaged ensemble predictions.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(60), x2 = rnorm(60))
df$y <- 2 * (df$x1 > 0) + rnorm(60, sd = 0.2)
bag_fit <- bagging_figs(y ~ x1 + x2, data = df, n_estimators = 3)
predict(bag_fit, new_data = df)
```

```
predict.figsr_fit
```

Predict Method for figsr_fit Models

Description

Predict Method for figsr_fit Models

Usage

```
## S3 method for class 'figsr_fit'
predict(object, new_data, type = NULL, ...)
```

Arguments

object	A fitted figsr_fit object.
new_data	A data frame of new predictor observations.
type	Character. Either "numeric", "class", or "prob". Default depends on model mode.
...	Additional arguments.

Value

A tibble with predictions in standardized tidymodels format.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(50), y = rnorm(50))
fit <- figs(y ~ x1, data = df)
predict(fit, new_data = df)
```

predict_figs	<i>Internal / Parsnip Predict Bridge Function for figsr_fit</i>
--------------	---

Description

Internal / Parsnip Predict Bridge Function for figsr_fit

Usage

```
predict_figs(object, new_data, type = "numeric", ...)
```

Arguments

object	A fitted figsr_fit object.
new_data	A data frame of new predictor observations.
type	Character. "numeric", "class", or "prob".
...	Additional arguments.

Value

A tibble of predictions.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(60))
df$y <- 2 * (df$x1 > 0) + rnorm(60, sd = 0.2)
fit <- figs(y ~ x1, data = df, max_splits = 3)
predict_figs(fit, new_data = df, type = "numeric")
```

print.figsr_fit	<i>Print Method for figsr_fit Models</i>
-----------------	--

Description

Print Method for figsr_fit Models

Usage

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

Arguments

x	A figsr_fit object.
...	Additional arguments, currently ignored.

Value

x, invisibly. Called for the side effect of printing a summary of the fitted tree sum to the console.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(60), x2 = rnorm(60))
df$y <- 2 * (df$x1 > 0) + rnorm(60, sd = 0.2)
print(figs(y ~ x1 + x2, data = df, max_splits = 3))
```

summary.figsr_fit	<i>Summary Method for figsr_fit Models</i>
-------------------	--

Description

summary.figsr_fit() prints human-readable decision rules for each tree in the sum.

Usage

```
## S3 method for class 'figsr_fit'
summary(object, ...)
```

Arguments

object	A figsr_fit object.
...	Additional arguments, currently ignored.

Value

object, invisibly. Called for the side effect of printing the IF-THEN decision rules of each tree to the console.

Examples

```
set.seed(42)
df <- data.frame(x1 = rnorm(60), x2 = rnorm(60))
df$y <- 2 * (df$x1 > 0) + rnorm(60, sd = 0.2)
summary(figs(y ~ x1 + x2, data = df, max_splits = 3))
```

update.figs_tree

*Update a FIGS Model Specification***Description**

update() changes the arguments of a `figs_tree()` specification in place, the way tidymodels users expect of any model specification.

Usage

```
## S3 method for class 'figs_tree'
update(
  object,
  max_splits = NULL,
  max_trees = NULL,
  min_n = NULL,
  fresh = FALSE,
  ...
)
```

Arguments

object	A <code>figs_tree()</code> specification.
max_splits	An integer for the maximum total splits across all trees.
max_trees	An integer for the maximum number of trees.
min_n	An integer for the minimum number of data points in a node to split.
fresh	Logical. Should the arguments be replaced rather than merged?
...	Not used.

Value

An updated `figs_tree()` specification.

Examples

```
spec <- figs_tree(max_splits = 4)
update(spec, max_splits = 8)
```

Index

- * **datasets**
 - pima_diabetes, [9](#)
- bagging_figs, [2](#)
- figs, [3](#)
- figs(), [3](#), [7](#)
- figs_tree, [5](#)
- figs_tree(), [7](#), [15](#)
- figsr_importance, [6](#)
- fit_figs, [7](#)
- max_splits, [8](#)
- max_trees, [9](#)
- pima_diabetes, [9](#)
- plot.figsr_fit, [10](#)
- predict.bagging_figs_fit, [12](#)
- predict.figsr_fit, [12](#)
- predict_figs, [13](#)
- print.figsr_fit, [14](#)
- stats::model.frame(), [4](#)
- stats::na.omit(), [4](#)
- summary.figsr_fit, [14](#)
- update.figs_tree, [15](#)