Package 'figsr'

October 28, 2026

Title Fast Interpretable Greedy-Tree Sums for Tree Ensembles

Version 0.2.0

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, 'ggplot2' graphics, bootstrap ensembling and seamless integration with 'parsnip' and 'tidymodels' workflows. The method is described in Tan et al. (2023) <doi:10.1073/pnas.2310151122>.

License MIT + file LICENSE

URL https://bonijoao.github.io/figsr/, https://github.com/bonijoao/figsr

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

Depends R (>= 4.1)

Imports dials, ggplot2 (>= 4.0.0), graphics, parsnip, rlang, stats, tibble

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

VignetteBuilder knitr, rmarkdown

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author João Paulo Assis Bonifácio [aut, cre, cph] (ORCID: <https://orcid.org/0009-0009-2335-8391>), Geraldo Magela da Cruz Pereira [aut, cph] (ORCID: <https://orcid.org/0000-0001-6280-4870>),

Pedro Mambelli Fernandes [aut, cph] (ORCID:
<https://orcid.org/0009-0001-1695-279X>),
João Vitor Andrade Alves de Souza [aut, cph] (ORCID:
<https://orcid.org/0009-0006-2780-318X>)

Maintainer João Paulo Assis Bonifácio <jpab.27@hotmail.com>

Repository CRAN

Date/Publication 2026-10-28 03:20:02 UTC

Contents

bagging_figs . . . 2
figs . . . 3
figsr_importance . . . 5
figs_tree . . . 6
fit_figs . . . 7
ggfigs . . . 8
max_splits . . . 10
max_trees . . . 10
pima_diabetes . . . 11
plot.figsr_fit . . . 12
predict.bagging_figs_fit . . . 13
predict.figsr_fit . . . 14
predict_figs . . . 15
print.figsr_fit . . . 16
summary.figsr_fit . . . 16
update.figs_tree . . . 17

Index **18**

`bagging_figs` *Bagging-FIGS: Ensemble of FIGS Models*

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

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)
```

---

figs *Fit Fast Interpretable Greedy-Tree Sums (FIGS)*

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

formula	A formula specifying outcome and predictor variables.
data	A data frame containing training data.
max_splits	Integer. Maximum total number of splits across all trees in the sum. Default is 10.
max_trees	Integer. Maximum number of trees allowed in the sum. Default is NULL (unconstrained up to max_splits).
min_n	Integer. Minimum number of observations required in a node to split. Default is 5.
mode	Character. Either "regression" or "classification". Only two-class outcomes are supported in classification mode. Default is "regression".
subset	An optional vector selecting the rows of data to fit on, passed to stats::model.frame().
na.action	A function describing what to do with missing values, passed to stats::model.frame(). Defaults to stats::na.omit(). Cannot be combined with na_method = "mia".
na_method	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 na.action.
...	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)
```

`figsr_importance` *Calculate Variable Importance for a* `figsr_fit` *Model*

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

`object`	A fitted `figsr_fit` model object.
`relative`	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)
```

---

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

`mode`	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.
`engine`	A single character string for the computational engine. Only `"figsr"` is available.
`max_splits`	An integer for the maximum total splits across all trees. `NULL` (the default) leaves the engine default of 10 in place.
`max_trees`	An integer for the maximum number of trees. `NULL` (the default) leaves the number of trees unconstrained.
`min_n`	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
```

`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

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

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)
```

`ggfigs` *'ggplot2' Graphics for* `figsr_fit` *Models*

Description

`ggfigs()` draws a fitted Fast Interpretable Greedy-Tree Sums (FIGS) model with 'ggplot2'. Five views are available:

- `"tree"`: every tree of the sum as a node-link diagram, edge width proportional to the observations that flow through it and leaves filled by their contribution on a diverging scale.
- `"rules"`: every tree as a horizontal list of rules, with + between trees. Each node continues to the right through the child with more nodes below it and the other child drops down, so a tree in which every split has a leaf child reads as a single line.
- `"waterfall"`: where the prediction for one observation comes from, as one bar per tree accumulating to the final prediction.
- `"cuts"`: every numeric cutpoint drawn over the data it was chosen from, one panel per predictor.
- `"profile"`: the exact partial dependence of the prediction on each predictor that carries a split.

With `interactive = FALSE` (the default) the result is an ordinary `ggplot` object, ready for `ggsave()`, PDF output or further + layers. With `interactive = TRUE` it is an HTML widget built with 'ggiraph', with tooltips that show the full rule behind each node, leaf or cut; this needs the 'ggiraph' package to be installed.

`autoplot()` is the same function under the 'ggplot2' generic, and always returns a `ggplot`.

Usage

```
ggfigs(
  x,
  type = c("tree", "rules", "waterfall", "cuts", "profile"),
  data = NULL,
  newdata = NULL,
  tree_idx = NULL,
  digits = 3,
  interactive = FALSE,
```

```
  ...
)

## S3 method for class 'figsr_fit'
autoplot(
  object,
  type = c("tree", "rules", "waterfall", "cuts", "profile"),
  ...
)
```

Arguments

x, object	A fitted figsr_fit model object.
type	The view to draw: "tree", "rules", "waterfall", "cuts" or "profile".
data	The data frame the model was fit on. Required by "cuts" and "profile", which draw the data, and optional for "rules", which then adds the outcome rate (classification) or mean (regression) to each rule. The fitted model does not store its training data.
newdata	A data frame with exactly one row, the observation that type = "waterfall" decomposes.
tree_idx	Optional integer vector of the trees to draw in type = "tree". Default NULL draws them all.
digits	Significant digits for cutpoints and leaf values.
interactive	Logical. FALSE (default) returns a ggplot; TRUE returns a 'ggiraph' girafe HTML widget.
...	Further arguments for the chosen view: width_svg and height_svg (inches) size the interactive widget; grid_n (default 160) and features select the grid and the predictors of "profile"; max_points (default 2000) caps the points drawn by "cuts".

Value

A ggplot object, or a girafe HTML widget when interactive = TRUE. For type = "rules" the ggplot carries an attribute "ggfigs_size", the natural c(width, height) of the drawing in inches, to size ggsave() with.

See Also

plot.figsr_fit() for the base-graphics tree diagram.

Examples

```
fit <- figs(mpg ~ wt + hp + disp, data = mtcars, max_splits = 5)
ggfigs(fit)
ggfigs(fit, type = "rules", data = mtcars)
ggfigs(fit, type = "waterfall", newdata = mtcars[1, ])
ggfigs(fit, type = "cuts", data = mtcars)
ggfigs(fit, type = "profile", data = mtcars)
```

```
# the 'ggplot2' generic gives the same plots
ggplot2::autoplot(fit, type = "rules")


if (requireNamespace("ggiraph", quietly = TRUE)) {
  ggfigs(fit, type = "rules", data = mtcars, interactive = TRUE)
}
```

`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` *Dials Parameter for Maximum Trees*

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 *Pima Indians Diabetes Database*

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, https://pmc.ncbi.nlm.nih.gov/articles/PMC2245318/. The data were long distributed by the UCI Machine Learning Repository.

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).

Nodes are spaced from the width of their own labels, so boxes and branch conditions never overlap, and each tree gets a panel width proportional to the room it needs. When a tree does not fit its panel, the whole drawing is scaled down – text included – rather than clipped; enlarge the device to get larger text.

Usage

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

Arguments

x	A fitted figsr_fit model object.
tree_idx	Optional integer vector specifying indices of trees to plot. Default is NULL (plots all trees).
style	Character string specifying the visual style: "scientific" (default, scientific/editorial notation), "modern" (rounded pastel boxes), or "classic" (high-contrast minimalist statistical style).
main	Optional character string. An overall title drawn once above every tree panel. Default is NULL (no overall title, the previous behavior).
tree_names	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 x$trees – indexed by each tree's position in the fit, not by plotting order, so it stays correct when tree_idx only plots some of them. NA in a given position falls back to the default title for that tree only. Default is NULL (the previous, automatic titles).
...	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` *Internal / Parsnip Predict Bridge Function for* `figsr_fit`

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 *Print Method for* figsr_fit *Models*

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 *Summary Method for* figsr_fit *Models*

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 `figs_tree()` 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, 11

autoplot.figsr_fit *(*ggfigs*)*, 8

bagging_figs, 2

figs, 3
figs(), *3*, *7*
figs_tree, 6
figs_tree(), *7*, *17*
figsr_importance, 5
fit_figs, 7

ggfigs, 8

max_splits, 10
max_trees, 10

pima_diabetes, 11
plot.figsr_fit, 12
plot.figsr_fit(), *9*
predict.bagging_figs_fit, 13
predict.figsr_fit, 14
predict_figs, 15
print.figsr_fit, 16

stats::model.frame(), *4*
stats::na.omit(), *4*
summary.figsr_fit, 16

update.figs_tree, 17