Package: parttree (via r-universe)

August 27, 2024

Title Visualise simple decision tree partitions

Version 0.0.1.9004

Description Simple functions for plotting 2D decision tree partition plots.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LazyData true

URL https://github.com/grantmcdermott/parttree, http://grantmcdermott.com/parttree

BugReports https://github.com/grantmcdermott/parttree/issues

Depends ggplot2 (>= 3.4.0)

Imports rpart, data.table, partykit, rlang

Suggests tinytest, palmerpenguins, titanic, mlr3, parsnip, workflows, magick, imager, patchwork, knitr, rmarkdown

VignetteBuilder knitr

Repository https://grantmcdermott.r-universe.dev

RemoteUrl https://github.com/grantmcdermott/parttree

RemoteRef HEAD

RemoteSha d2b60ac06bc760751685a9e92ff590f731f6a0ed

Contents

geom_parttree *Visualise tree partitions*

Description

geom_parttree() is a simple extension of ggplot2::geom_rect()that first calls parttree() to convert the inputted tree object into an amenable data frame.

Usage

```
geom_parttree(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  flipaxes = FALSE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	An rpart::rpart.object or an object of compatible type (e.g. a decision tree constructed via the partykit, tidymodels, or mlr3 front-ends).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
linejoin	Line join style (round, mitre, bevel).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

flipaxes	Logical. By default, the "x" and "y" axes variables for plotting are determined by the first split in the tree. This can cause plot orientation mismatches depending on how users specify the other layers of their plot. Setting to TRUE will flip the "x" and "y" variables for the geom_parttree layer.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

Details

Because of the way that ggplot2 validates inputs and assembles plot layers, note that the data input for geom_parttree() (i.e. decision tree object) must assigned in the layer itself; not in the initialising ggplot2::ggplot() call. See Examples.

Aesthetics

geom_parttree() aims to "work-out-of-the-box" with minimal input from the user's side, apart from specifying the data object. This includes taking care of the data transformation in a way that, generally, produces optimal corner coordinates for each partition (i.e. xmin, xmax, ymin, and ymax). However, it also understands the following aesthetics that users may choose to specify manually:

- fill (particularly encouraged, since this will provide a visual cue regarding the prediction in each partition region)
- colour
- alpha
- linetype
- size

See Also

parttree(), ggplot2::geom_rect().

Examples

```
library(rpart)

### Simple decision tree (max of two predictor variables)

iris_tree = rpart(Species ~ Petal.Length + Petal.Width, data=iris)

## Plot with original iris data only
p = ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width)) +
  geom_point(aes(col = Species))

## Add tree partitions to the plot (borders only)
p + geom_parttree(data = iris_tree)

## Better to use fill and highlight predictions
p + geom_parttree(data = iris_tree, aes(fill = Species), alpha=0.1)
```

```
## To drop the black border lines (i.e. fill only)
p + geom_parttree(data = iris_tree, aes(fill = Species), col = NA, alpha = 0.1)


### Example with plot orientation mismatch

p2 = ggplot(iris, aes(x=Petal.Width, y=Petal.Length)) +
  geom_point(aes(col=Species))

## Oops
p2 + geom_parttree(data = iris_tree, aes(fill=Species), alpha = 0.1)

## Fix with 'flipaxes = TRUE'
p2 + geom_parttree(data = iris_tree, aes(fill=Species), alpha = 0.1, flipaxes = TRUE)


### Various front-end frameworks are also supported, e.g.:

library(parsnip)

iris_tree_parsnip =
  decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification") %>%
  fit(Species ~ Petal.Length + Petal.Width, data=iris)

p + geom_parttree(data = iris_tree_parsnip, aes(fill=Species), alpha = 0.1)


### Trees with continuous independent variables are also supported. But you
### may need to adjust (or switch off) the fill legend to match the original
### data, e.g.:

iris_tree_cont = rpart(Petal.Length ~ Sepal.Length + Petal.Width, data=iris)
p3 = ggplot(data = iris, aes(x = Petal.Width, y = Sepal.Length)) +
 geom_parttree(
   data = iris_tree_cont,
   aes(fill = Petal.Length), alpha=0.5
   ) +
  geom_point(aes(col = Petal.Length)) +
  theme_minimal()

## Legend scales don't quite match here:
p3

## Better to scale fill to the original data
p3 + scale_fill_continuous(limits = range(iris$Petal.Length))
```

---

parttree *Convert a decision tree into a data frame of partition coordinates*

Description

Extracts the terminal leaf nodes of a decision tree with one or two numeric predictor variables. These leaf nodes are then converted into a data frame, where each row represents a partition (or leaf or terminal node) that can easily be plotted in coordinate space.

Usage

```
parttree(tree, keep_as_dt = FALSE, flipaxes = FALSE)
```

Arguments

`tree`	A tree object. Supported classes include rpart::rpart.object, or the compatible classes from from the `parsnip`, `workflows`, or `mlr3` front-ends, or the `constparty` class inheriting from `partykit::party()`.
`keep_as_dt`	Logical. The function relies on `data.table` for internal data manipulation. But it will coerce the final return object into a regular data frame (default behavior) unless the user specifies `TRUE`.
`flipaxes`	Logical. The function will automatically set the y-axis variable as the first split variable in the tree provided unless the user specifies `TRUE`.

Details

This function can be used with a regression or classification tree containing one or (at most) two numeric predictors.

Value

A data frame comprising seven columns: the leaf node, its path, a set of coordinates understandable to `ggplot2` (i.e., xmin, xmax, ymin, ymax), and a final column corresponding to the predicted value for that leaf.

See Also

`geom_parttree()`, `rpart::rpart()`, `partykit::ctree()`.

Examples

```
## rpart trees
library("rpart")
rp = rpart(Species ~ Petal.Length + Petal.Width, data = iris)
parttree(rp)

## conditional inference trees
library("partykit")
ct = ctree(Species ~ Petal.Length + Petal.Width, data = iris)
parttree(ct)

## rpart via partykit
rp2 = as.party(rp)
parttree(rp2)
```

Index