

Package ‘ggparty’

July 10, 2025

Title 'ggplot' Visualizations for the 'partykit' Package

Version 1.0.0.1

Copyright file inst/COPYRIGHTS

Description Extends 'ggplot2' functionality to the 'partykit' package. 'ggparty' provides the necessary tools to create clearly structured and highly customizable visualizations for tree-objects of the class 'party'.

Maintainer Martin Borkovec <martin.borkovec@skyforge.at>

Depends R (>= 3.4.0), ggplot2, partykit

Imports grid, gtable, utils, checkmate, methods, survival, rlang

Suggests testthat, mlbench, AER, coin, vdiff, knitr, rmarkdown,
pander, MASS, TH.data

License GPL-2 | GPL-3

URL <https://github.com/martin-borkovec/ggparty>

BugReports <https://github.com/martin-borkovec/ggparty/issues>

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Martin Borkovec [aut, cre],
Niyaz Madin [aut],
Hadley Wickham [ctb],
Winston Chang [ctb],
Lionel Henry [ctb],
Thomas Lin Pedersen [ctb],
Kohske Takahashi [ctb],
Claus Wilke [ctb],
Kara Woo [ctb],
Hiroaki Yutani [ctb]

Repository CRAN

Date/Publication 2025-07-10 16:31:21 UTC

Contents

autoplot.party	2
geom_edge	3
geom_edge_label	4
geom_node_label	6
geom_node_plot	9
get_predictions	12
ggparty	12
makeContent.nodeplotgrob	14

Index	15
-------	----

autoplot.party	<i>autoplot methods for party objects</i>
----------------	---

Description

autoplot methods for party objects

Usage

```
## S3 method for class 'party'
autoplot(object, ...)

## S3 method for class 'constparty'
autoplot(object, ...)

## S3 method for class 'modelparty'
autoplot(object, plot_var = NULL, ...)

## S3 method for class 'lmtree'
autoplot(object, plot_var = NULL, show_fit = TRUE, ...)
```

Arguments

object	object of class party.
...	additional parameters
plot_var	Which covariate to plot against response. Defaults to second column in ‘data’ of tree.
show_fit	If TRUE ‘fitted_values’ are drawn.

Examples

```
library(ggparty)

data("WeatherPlay", package = "partykit")
sp_o <- partysplit(1L, index = 1:3)
sp_h <- partysplit(3L, breaks = 75)
sp_w <- partysplit(4L, index = 1:2)
pn <- partynode(1L, split = sp_o, kids = list(
  partynode(2L, split = sp_h, kids = list(
    partynode(3L, info = "yes"),
    partynode(4L, info = "no"))),
  partynode(5L, info = "yes"),
  partynode(6L, split = sp_w, kids = list(
    partynode(7L, info = "yes"),
    partynode(8L, info = "no")))))
py <- party(pn, WeatherPlay)

autoplot(py)
```

geom_edge

*Draw edges***Description**

Draws edges between children and parent nodes. Wrapper for [ggplot2::geom_segment\(\)](#)

Usage

```
geom_edge(
  mapping = NULL,
  nudge_x = 0,
  nudge_y = 0,
  ids = NULL,
  show.legend = NA,
  ...
)
```

Arguments

mapping	Mapping of x, y, xend and yend defaults to ids' and their parent's coordinates. Other mappings can be added here as aes() .
nudge_x, nudge_y	Nudge labels.
ids	Choose which edges to draw by their children's ids.
show.legend	logical See ggplot2::layer() .
...	Additional arguments for ggplot2::geom_segment() .

See Also

[ggparty\(\)](#), [geom_edge\(\)](#)

Examples

```
library(ggparty)
data("WeatherPlay", package = "partykit")
sp_o <- partysplit(1L, index = 1:3)
sp_h <- partysplit(3L, breaks = 75)
sp_w <- partysplit(4L, index = 1:2)
pn <- partynode(1L, split = sp_o, kids = list(
  partynode(2L, split = sp_h, kids = list(
    partynode(3L, info = "yes"),
    partynode(4L, info = "no"))),
  partynode(5L, info = "yes"),
  partynode(6L, split = sp_w, kids = list(
    partynode(7L, info = "yes"),
    partynode(8L, info = "no")))))
py <- party(pn, WeatherPlay)

ggparty(py) +
  geom_edge() +
  geom_edge_label() +
  geom_node_label(aes(label = splitvar),
    ids = "inner") +
  geom_node_label(aes(label = info),
    ids = "terminal")
```

geom_edge_label

Draw edge labels

Description

Label edges with corresponding split breaks

Usage

```
geom_edge_label(
  mapping = NULL,
  nudge_x = 0,
  nudge_y = 0,
  ids = NULL,
  shift = 0.5,
  label.size = 0,
  splitlevels = seq_len(100),
  max_length = NULL,
  parse_all = FALSE,
  parse = TRUE,
  ...
)
```

Arguments

mapping	Mapping of label defaults to breaks_label . Other mappings can be added here as <code>aes()</code> .
nudge_x, nudge_y	Nudge label.
ids	Choose which splitbreaks to label by their children's ids.
shift	Value in (0,1). Moves label along corresponding edge.
label.size	See <code>ggplot2::geom_label()</code> .
splitlevels	Which levels of split to plot. This may be useful in the presence of many factor levels for one split break.
max_length	If provided breaks_label levels will be truncated to the specified length.
parse_all	Defaults to FALSE, in which case everything but the inequality signs of breaks_label are deparsed. If TRUE complete breaks_label are parsed.
parse	Needs to be true in order to parse inequality signs of breaks_label .
...	Additional arguments for <code>ggplot2::geom_label()</code> .

See Also

[ggparty\(\)](#)

Examples

```
library(ggparty)
data("WeatherPlay", package = "partykit")
sp_o <- partysplit(1L, index = 1:3)
sp_h <- partysplit(3L, breaks = 75)
sp_w <- partysplit(4L, index = 1:2)
pn <- partynode(1L, split = sp_o, kids = list(
  partynode(2L, split = sp_h, kids = list(
    partynode(3L, info = "yes"),
    partynode(4L, info = "no"))),
  partynode(5L, info = "yes"),
  partynode(6L, split = sp_w, kids = list(
    partynode(7L, info = "yes"),
    partynode(8L, info = "no")))))
py <- party(pn, WeatherPlay)

ggparty(py) +
  geom_edge() +
  geom_edge_label() +
  geom_node_label(aes(label = splitvar),
    ids = "inner") +
  geom_node_label(aes(label = info),
    ids = "terminal")
```

geom_node_label	<i>Draw (multi-line) labels at nodes</i>
-----------------	--

Description

geom_node_splitvar() and geom_node_info() are simplified versions of geom_node_label() with the respective defaults to either label the split variables for all inner nodes or the info for all terminal nodes.

Usage

```
geom_node_label(
  mapping = NULL,
  data = NULL,
  line_list = NULL,
  line_gpar = NULL,
  ids = NULL,
  position = "identity",
  ...,
  parse = FALSE,
  nudge_x = 0,
  nudge_y = 0,
  label.padding = unit(0.25, "lines"),
  label.r = unit(0.15, "lines"),
  label.size = 0.25,
  label.col = NULL,
  label.fill = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_node_info(
  mapping = NULL,
  nudge_x = 0,
  nudge_y = 0,
  ids = NULL,
  label.padding = unit(0.5, "lines"),
  ...
)

geom_node_splitvar(
  mapping = NULL,
  nudge_x = 0,
  nudge_y = 0,
  label.padding = unit(0.5, "lines"),
  ids = NULL,
```

```
    ...  
  )
```

Arguments

mapping	x and y are mapped per default to the node's coordinates. If you don't want to set line specific graphical parameters, you can also map label here. Otherwise set labels in line_list.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
line_list	Use this only if you want a multi-line label with the possibility to override the aesthetics mapping for each line specifically with fixed graphical parameters. In this case, don't map anything to label in the <code>aes()</code> supplied to mapping, but instead pass here a list of <code>aes()</code> with the only mapped variable in each being label. Other aesthetic mappings still can be passed to mapping and will apply to all lines and the border, unless overwritten by line_gpar. The order of the list represents the order of the plotted lines.
line_gpar	List of lists containing line-specific graphical parameters. Only use in conjunction with line_list. Has to contain the same number of lists as are <code>aes()</code> in line_list. First list applies to first line, and so on.
ids	Select for which nodes to draw a label. Can be "inner", "terminal", "all" or numeric vector of ids.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Additional arguments to layer.
parse	If TRUE, the labels will be parsed into expressions. Can also be specified per line via line_gpar.
nudge_x, nudge_y	Adjust position of label.
label.padding	Amount of padding around label. Defaults to 0.25 lines.

<code>label.r</code>	Radius of rounded corners. Defaults to 0.15 lines.
<code>label.size</code>	Size of label border, in mm.
<code>label.col</code>	Border colour.
<code>label.fill</code>	Background colour.
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	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.
<code>inherit.aes</code>	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. <code>borders()</code> .

Details

`geom_node_label()` is a modified version of `ggplot2::geom_label()`. This modification allows for labels with multiple lines and line specific graphical parameters.

See Also

[ggparty\(\)](#)

Examples

```
library(ggparty)
data("WeatherPlay", package = "partykit")
sp_o <- partysplit(1L, index = 1:3)
sp_h <- partysplit(3L, breaks = 75)
sp_w <- partysplit(4L, index = 1:2)
pn <- partynode(1L, split = sp_o, kids = list(
  partynode(2L, split = sp_h, kids = list(
    partynode(3L, info = "yes"),
    partynode(4L, info = "no"))),
  partynode(5L, info = "yes"),
  partynode(6L, split = sp_w, kids = list(
    partynode(7L, info = "yes"),
    partynode(8L, info = "no")))))
py <- party(pn, WeatherPlay)

ggparty(py) +
  geom_edge() +
  geom_edge_label() +
  geom_node_label(aes(label = splitvar),
    ids = "inner") +
  geom_node_label(aes(label = info),
    ids = "terminal")

#####
```



```

data("TeachingRatings", package = "AER")
tr <- subset(TeachingRatings, credits == "more")

tr_tree <- lmtree(eval ~ beauty | minority + age + gender + division + native +
                  tenure, data = tr, weights = students, caseweights = FALSE)

data("TeachingRatings", package = "AER")
tr <- subset(TeachingRatings, credits == "more")

tr_tree <- lmtree(eval ~ beauty | minority + age + gender + division + native +
                  tenure, data = tr, weights = students, caseweights = FALSE)

ggparty(tr_tree,
        terminal_space = 0.5,
        add_vars = list(p.value = "$node$info$p.value")) +
geom_edge(size = 1.5) +
geom_edge_label(colour = "grey", size = 6) +
geom_node_plot(gglist = list(geom_point(aes(x = beauty,
                                             y = eval,
                                             col = tenure,
                                             shape = minority),
                                     alpha = 0.8),
                             theme_bw(base_size = 15)),
              scales = "fixed",
              id = "terminal",
              shared_axis_labels = TRUE,
              shared_legend = TRUE,
              legend_separator = TRUE,
              predict = "beauty",
              predict_gpar = list(col = "blue",
                                  size = 1.2)
) +
geom_node_label(aes(col = splitvar),
               line_list = list(aes(label = paste("Node", id)),
                               aes(label = splitvar),
                               aes(label = paste("p =", formatC(p.value,
                                                                format = "e", digits = 2)))),
               line_gpar = list(list(size = 12, col = "black", fontface = "bold"),
                               list(size = 20),
                               list(size = 12)),
               ids = "inner") +
geom_node_label(aes(label = paste0("Node ", id, ", N = ", nodesize)),
               fontface = "bold",
               ids = "terminal",
               size = 5,
               nudge_y = 0.01) +
theme(legend.position = "none")

```

Description

Additional component for a `[ggparty()]` that allows to create in each node a `ggplot` with its data. `#'`

Usage

```
geom_node_plot(
  plot_call = "ggplot",
  gglist = NULL,
  width = 1,
  height = 1,
  size = 1,
  ids = "terminal",
  scales = "fixed",
  nudge_x = 0,
  nudge_y = 0,
  shared_axis_labels = FALSE,
  shared_legend = TRUE,
  predict = NULL,
  predict_gpar = NULL,
  legend_separator = FALSE
)
```

Arguments

<code>plot_call</code>	Any function that generates a 'ggplot2' object.
<code>gglist</code>	List of additional 'gg' components. Columns of 'data' of nodes can be mapped. Additionally 'fitted_values' and 'residuals' can be mapped if present in 'party' of 'ggparty()'
<code>width</code>	Expansion factor for viewport's width.
<code>height</code>	Expansion factor for viewport's height.
<code>size</code>	Expansion factor for viewport's size.
<code>ids</code>	Id's to plot. Numeric, "terminal", "inner" or "all". Defaults to "terminal".
<code>scales</code>	See <code>[ggplot2::facet_wrap()]</code>
<code>nudge_x</code> , <code>nudge_y</code>	Nudges node plot.
<code>shared_axis_labels</code>	If TRUE only one pair of axes labels is plotted in the terminal space. Only recommended if 'ids' "terminal" or "all".
<code>shared_legend</code>	If 'TRUE' one shared legend is plotted at the bottom of the tree.
<code>predict</code>	Character string specifying variable for which predictions should be plotted.
<code>predict_gpar</code>	Named list containing arguments to be passed to the 'geom_line()' call of predicted values.
<code>legend_separator</code>	If 'TRUE' line between legend and tree is drawn.

See Also

[ggparty()]

Examples

```
library(ggparty)

airq <- subset(airquality, !is.na(Ozone))
airct <- ctree(Ozone ~ ., data = airq)

ggparty(airct, horizontal = TRUE, terminal_space = 0.6) +
  geom_edge() +
  geom_edge_label() +
  geom_node_splitvar() +
  geom_node_plot(gglist = list(
    geom_density(aes(x = Ozone))),
    shared_axis_labels = TRUE)

#####

## Plot with ggparty

## Demand for economics journals data
data("Journals", package = "AER")
Journals <- transform(Journals,
  age = 2000 - foundingyear,
  chars = charpp * pages)

## linear regression tree (OLS)
j_tree <- lmtree(log(subs) ~ log(price/citations) | price + citations +
  age + chars + society, data = Journals, minsize = 10, verbose = TRUE)

pred_df <- get_predictions(j_tree, ids = "terminal", newdata = function(x) {
  data.frame(
    citations = 1,
    price = exp(seq(from = min(x$log(price/citations)),
      to = max(x$log(price/citations)),
      length.out = 100)))
})

ggparty(j_tree, terminal_space = 0.8) +
  geom_edge() +
  geom_edge_label() +
  geom_node_splitvar() +
  geom_node_plot(gglist =
    list(aes(x = log(price/citations), y = log(subs)),
      geom_point(),
      geom_line(data = pred_df,
        aes(x = log(price/citations),
          y = prediction),
          col = "red"))))
```

get_predictions	Create data.frame with predictions for each node
-----------------	--

Description

Create data.frame with predictions for each node

Usage

```
get_predictions(party_object, ids, newdata_fun, predict_arg = NULL)
```

Arguments

party_object	object of class 'party'
ids	Id's to plot. Numeric, "terminal", "inner" or "all". MUST be identical to 'ids' of [geom_node_plot()] used to plot this data.
newdata_fun	function which takes 'data' of node and returns 'newdata' for 'predict()'
predict_arg	list of additional arguments passed to [predict()]

ggparty	Create a new ggparty plot
---------	---------------------------

Description

'ggplot2' extension for objects of class 'party'. Creates a 'data.frame' from an object of class 'party' and calls [ggplot2::ggplot()]

Usage

```
ggparty(
  party,
  horizontal = FALSE,
  terminal_space,
  layout = NULL,
  add_vars = NULL
)
```

Arguments

party	Object of class 'party'.
horizontal	If 'TRUE' plot will be horizontal.
terminal_space	Proportion of the plot that should be reserved for the terminal nodeplots. Defaults to '2 / (depth(party) + 2)'.

layout	Optional layout adjustment. Overwrites the coordinates of the specified nodes. Must be 'data.frame' containing the columns 'id', 'x' and 'y'. With 'x' and 'y' values between 0 and 1.
add_vars	Named list containing either string(s) specifying the locations of elements to be extracted from each node of 'party' or function(s) of corresponding row of plot data and node. In either case returned object has to be of length 1. If the data is supposed to be accessible by [geom_node_plot()] the respective list entry has to be named with the prefix "nodedata_" and be a function returning a list of same length as 'nodesize'.

Details

'ggparty' can be called directly with an object of class 'party', which will convert it to a suitable 'data.frame' and pass it to a call to 'ggplot' with as the 'data' argument. As usual, additional components can then be added with '+'.

The nodes will be spaced equally in the unit square. Specifying 'terminal_size' allows to increase or decrease the area for plots of the terminal nodes.

If one of the list entries supplied to 'add_vars' is a function, it has to take exactly two arguments, namely 'data' (the corresponding row of the plot_data data frame) and 'node' (the corresponding node, i.e. 'party_object[i]')

See Also

[geom_edge()], [geom_edge_label()], [geom_node_label()], [autoplot.party()], [geom_node_plot()]

Examples

```
library(ggparty)
data("WeatherPlay", package = "partykit")
sp_o <- partysplit(1L, index = 1:3)
sp_h <- partysplit(3L, breaks = 75)
sp_w <- partysplit(4L, index = 1:2)
pn <- partynode(1L, split = sp_o, kids = list(
  partynode(2L, split = sp_h, kids = list(
    partynode(3L, info = "yes"),
    partynode(4L, info = "no"))),
  partynode(5L, info = "yes"),
  partynode(6L, split = sp_w, kids = list(
    partynode(7L, info = "yes"),
    partynode(8L, info = "no")))))
py <- party(pn, WeatherPlay)

ggparty(py) +
  geom_edge() +
  geom_edge_label() +
  geom_node_label(aes(label = splitvar,
    ids = "inner")) +
  geom_node_label(aes(label = info,
    ids = "terminal"))
```

makeContent.nodeplotgrob

apparently needs to be exported

Description

apparently needs to be exported

Usage

```
## S3 method for class 'nodeplotgrob'  
makeContent(x)
```

Arguments

x nodeplotgrob

Index

`autoplot.constparty (autoplot.party)`, [2](#)
`autoplot.lmtree (autoplot.party)`, [2](#)
`autoplot.modelparty (autoplot.party)`, [2](#)
`autoplot.party`, [2](#)

`borders()`, [8](#)

`fortify()`, [7](#)

`geom_edge`, [3](#)
`geom_edge()`, [4](#)
`geom_edge_label`, [4](#)
`geom_node_info (geom_node_label)`, [6](#)
`geom_node_label`, [6](#)
`geom_node_plot`, [9](#)
`geom_node_splitvar (geom_node_label)`, [6](#)
`get_predictions`, [12](#)
`ggparty`, [12](#)
`ggparty()`, [4](#), [5](#), [8](#)
`ggplot()`, [7](#)
`ggplot2::geom_label()`, [5](#), [8](#)
`ggplot2::geom_segment()`, [3](#)
`ggplot2::layer()`, [3](#)

`layer position`, [7](#)

`makeContent.nodeplotgrob`, [14](#)