Package ‘ggtreeExtra’

May 26, 2022

Type Package

Title An R Package To Add Geometric Layers On Circular Or Other Layout Tree Of ``ggtree''

Version 1.7.0

Description 'ggtreeExtra' extends the method for mapping and visualizing associated data on phylogenetic tree using 'ggtree'. These associated data can be presented on the external panels to circular layout, fan layout, or other rectangular layout tree built by 'ggtree' with the grammar of 'ggplot2'.

Imports ggplot2, utils, rlang, ggnewscale, stats, ggtree, tidytree (>= 0.3.9)

Suggests treeio, ggstar, patchwork, knitr, rmarkdown, prettydoc, markdown, testthat (>= 3.0.0), pillar

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/YuLab-SMU/ggtreeExtra/

BugReports https://github.com/YuLab-SMU/ggtreeExtra/issues

ByteCompile true

Roxygen list(markdown = TRUE)

biocViews Software, Visualization, Phylogenetics, Annotation

RoxygenNote 7.1.2

VignetteBuilder knitr

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/ggtreeExtra

git_branch master

git_last_commit 800da3d

git_last_commit_date 2022-04-26

Date/Publication 2022-05-26

Author Shuangbin Xu [aut, cre] (<https://orcid.org/0000-0003-3513-5362>), Guangchuang Yu [aut, ctb] (<https://orcid.org/0000-0002-6485-8781>)

Maintainer Shuangbin Xu <xshuangbin@163.com>

R topics documented:

geom_fruit 2
geom_fruit_list 6
PositionDodgex 8
PositionJitterdodgex 8
PositionJitterx 8
position_dodgex 9
position_identityx 10
position_jitterdodgex 11
position_jitterx 12
position_points_jitterx 13
position_points_sinax 14
position_raincloudx 15
position_stackx 16

Index **18**

`geom_fruit` *plot tree with associated data in another method.*

Description

'geom_fruit()' can automatically re-arrange the input 'data' according to the tree structure. It can present the associated data on the external panels of the tree using the 'geom' function defined in 'ggplot2' or other ggplot2-based packages with aesthetic 'mapping' and other parameters, and it will align the external layers in the outer ring of circular layout tree or with rectangular layout tree side by side. Note: the tree should be created by 'ggtree'.

Usage

```
geom_fruit(
  mapping,
  data = NULL,
  geom,
  offset = 0.03,
  pwidth = 0.2,
  position = "auto",
  inherit.aes = FALSE,
  grid.params = NULL,
 axis.params = list(axis = "none", text.angle = 0, text.size = 0.8, text = NULL, title
  = NULL, title.size = 3, title.height = 0.1, title.angle = 0, title.color = "black",
    nbreak = 4, line.size = 0.2, line.color = "grey", line.alpha = 1, ...),
  ...
)

fruit_plot(
  p,
```

```
  data = NULL,
  geom,
  mapping,
  offset = 0.03,
  pwidth = 0.2,
  position = "auto",
  ...
)
```

Arguments

`mapping` aes mapping for 'geom'

`data` data to plot by 'geom', the column contained tree tip labels should be as y in mapping.

`geom` geom function to plot the data.

`offset` numeric, distance between external layers or between tree and external layers, default is 0.03, meaning the 0.03 times of x range of tree (0.03 * xrange of tree).

`pwidth` numeric, the width of external geometric layer, default is 0.2, meaning the 0.2 times of x range of tree (0.2 * xrange of tree).

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function, default is 'auto', see details in the following.

`inherit.aes` logical, 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, default is FALSE.

`grid.params` list, the parameters to control the attributes of grid lines, default is NULL, see the grid.params in the following.

grid.params control the attributes of grid line of external layer, it can be referred to the following parameters: ##'

- `vline` logical, whether add the vertical line, default is FALSE.
- `color` color of line, default is grey.
- `size` the width of line, default is 0.2.
- `alpha` the colour transparency of line, default is 1.
- `lineend` Line end style (round, butt, square), default is "butt".
- `linejoin` Line end style (round, butt, square), default is "round".
- `linetype` Type of line, default is 1.

`axis.params` list, the parameters to control the attributes of pseudo axis, see the axis.params in the following.

axis.params control the attributes of axis, it can be referred to the following parameters:

- `axis` character, add the axis, if it is set to "none", meaning don't display axis (default), "x" display the x axis, "y" display the y axis, "xy" display the two axis.
- `text` vector, the text of axis x, default is NULL, it is only valid when the text of axis is single and x is discrete.

- `vjust` numeric, A numeric specifying vertical justification, default is 0.5.
- `hjust` numeric, A numeric specifying horizontal justification, default is 0.5.
- `text.angle` numeric, the angle of axis text, default is 0.
- `text.size` numeric, the size of axis text, default is 0.8.
- `title` character, the title of panel or x-axis label, default is NULL, it is only valid when "x" axis exists.
- `title.size` numeric, the size of title text, default is 3.
- `title.height` numeric, the height of title text position more than tree, default is 0.1, it is relative to height of tree.
- `title.angle` numeric, the angle of title text, default is 0.
- `title.color` character, the color of title text, default is "black".
- `nbreak` numeric, meaning the number of axis to break, integer giving the *desired* number of intervals. Non-integer values are rounded down. It is only valid when x is continuous, default is 4.
- `line.size` numeric, the size of axis line, default is 0.2.
- `line.color` character, the color of axis line color, default is "grey".
- `line.alpha` numeric, the colour transparency of axis line, default is 1.

`...` additional parameters for 'geom'

`p` tree view

Details

The 'data' parameter is data.frame or tibble type, it is the same with the data for corresponding geometric layers, but it must contain one column of taxa labels of tree, it will be mapped to 'y' axis in 'mapping'. When 'data' is not provided, the associated data in tree data will be extracted automatically, and the 'y' axis don't need to be mapped.

The 'mapping' parameter is setting of aesthetic mappings created by 'aes()' or 'aes_()' of 'ggplot2', the 'y' should be assigned to the variable names of column of taxa labels in data.frame of 'data', only if the 'data' is not provided, see the above.

The 'geom' parameter is the geometric function defined in 'ggplot2' or other 'ggplot2-extension', e.g.

ggplot2	geom_bar,geom_col,geom_boxplot,geom_violin,geom_tile	circular, rectangular
ggmsa	geom_msa	rectangular
ggstar	geom_star	circular, rectangular
ggimage	geom_image,geom_phylopic	circular, rectangular
ggpmisc	geom_plot,geom_table	circular, rectangular
ggridges	geom_density_ridges	circular, rectangular
ggtext	geom_richtext	circular, rectangular
...		

if the 'geom' is 'geom_bar', 'geom_col', 'geom_boxplot', 'geom_violin', the 'orientation' should be specified to 'y'.

The default 'position' parameter is 'auto', it will guess and determine (hopefully) a suitable position for the specified geometric layer. That means using 'position_stackx()' for geom_bar(), 'po-

sition_dodgex()' for 'geom_violin()' and 'geom_boxplot()', and 'position_identityx()' for others (e.g. geom_point() and geom_tile() etc.). A geometric layer that has a position parameter should be compatible with geom_fruit(), as it allows using position functions defined in the ggtreeExtra package to adjust output layer position.

and the grid line also can be added using 'grid.params=list(...)'.

The axis line and text can be added using 'axis.params=list(axis="x",...)'.

The 'p' parameter only work when you use `fruit_plot()`, which is alias of `geom_fruit()`.

Value

ggplot object

Author(s)

Shuangbin Xu and Guangchuang Yu

Examples

```
library(ggtree)
library(ggplot2)
library(ggstar)
set.seed(1024)
tr <- rtree(100)
dd = data.frame(id=tr$tip.label, value=abs(rnorm(100)))
dt = data.frame(id=tr$tip.label, group=c(rep("A",50),rep("B",50)))
p <- ggtree(tr, layout="circular")

p1 <- p +
      geom_fruit(
          data=dt,
          geom=geom_star,
          mapping=aes(y=id, fill=group),
          size=2.5,
          starstroke=0
      )
p2 <- p1 +
      geom_fruit(
          data=dd,
          geom=geom_bar,
          mapping=aes(x=value, y=id),
          orientation="y",
          stat="identity"
      )

p3 <- fruit_plot(
           p=p,
           data=dt,
           geom=geom_star,
           mapping=aes(y=id, fill=group),
           size=2.5,
           starstroke=0
```

```
      )

p4 <- fruit_plot(
           p=p3,
           data=dd,
           geom=geom_bar,
           mapping=aes(x=value, y=id),
           orientation="y",
           stat="identity"
      )
p <- p %<+% dd %<+% dt
p5 <- p +
      geom_fruit(
          geom = geom_star,
          mapping = aes(y=id, fill=group),
          size = 2.5,
          starstroke = 0
      ) +
      geom_fruit(
          geom = geom_bar,
          mapping = aes(x=value, y=id),
          orientation = "y",
          stat = 'identity'
      )
```

`geom_fruit_list` *geom_fruit_list*

Description

add the layers to the same position out of ggtree.

Usage

```
geom_fruit_list(fruit, ...)
```

Arguments

`fruit`	the layer of geom_fruit.
`...,`	another layers of geom_fruit, or scales.

Value

ggplot object

Author(s)

Shuangbin Xu and GuangChuang Yu

Examples

```
library(ggplot2)
library(ggtree)
library(ggstar)
library(ggnewscale)
set.seed(1024)
tr <- rtree(100)
dt <- data.frame(id=tr$tip.label, value=abs(rnorm(100)), group=c(rep("A",50),rep("B",50)))
df <- dt
dtf <- dt
colnames(df)[[3]] <- "group2"
colnames(dtf)[[3]] <- "group3"

p <- ggtree(tr, layout="fan", open.angle=0)
# first circle
p1 <- p +
      geom_fruit(
          data=dt,
          geom=geom_bar,
          mapping=aes(y=id, x=value, fill=group),
          orientation="y",
          stat="identity"
      ) +
      new_scale_fill()
# second circle
fruitlist <- geom_fruit_list(
                 geom_fruit(
                     data = df,
                     geom = geom_bar,
                     mapping = aes(y=id, x=value, fill=group2),
                     orientation = "y",
                     stat = "identity"
                 ),
                 scale_fill_manual(values=c("blue", "red")),
                 new_scale_fill(),
                 geom_fruit(
                     data = dt,
                     geom = geom_star,
                     mapping = aes(y=id, x=value, fill=group),
                     size = 2.5,
                     color = NA,
                     starstroke = 0
                 )
             )

p2 <- p1 + fruitlist + new_scale_fill()
# third circle
p3 <- p2 +
      geom_fruit(
          data=dtf,
          geom=geom_bar,
          mapping = aes(y=id, x=value, fill=group3),
```

```
            orientation = "y",
            stat = "identity"
        ) +
        scale_fill_manual(values=c("#00AED7", "#009E73"))
p3
```

`PositionDodgex` *PositionDodgex*

Description

PositionDodgex

PositionDodgex2

PositionIdentityx

PositionPointsJitterx

PositionRaincloudx

PositionPointsSinax

PositionStackx

Author(s)

Shuangbin Xu

`PositionJitterdodgex` *PositionJitterdodgex*

Description

PositionJitterdodgex

`PositionJitterx` *PositionJitterx*

Description

PositionJitterx

`position_dodgex` *Dodge overlapping objects side-to-side which can be shifted verically or horizontally.*

Description

Dodging preserves the vertical position of an geom while adjusting the horizontal position. `position_dodgex2` is a special case of `position_dodgex` for arranging box plots, which can have variable widths. `position_dodgex2` also works with bars and rectangles. But unlike `position_dodgex`, `position_dodgex2` works without a grouping variable in a layer.

Usage

```
position_dodgex(
  width = NULL,
  hexpand = NA,
  vexpand = NA,
  preserve = c("total", "single")
)

position_dodgex2(
  width = NULL,
  preserve = c("total", "single"),
  hexpand = NA,
  vexpand = NA,
  padding = 0.1,
  reverse = FALSE
)
```

Arguments

`width`	Dodging width, when different to the width of the individual elements. This is useful when you want to align narrow geoms with wider geoms.
`hexpand`	numeric, Horizon expand for geoms that have a position, default is NA.
`vexpand`	numeric, Vertical expand for geoms that have a position, default is NA.
`preserve`	Should dodging preserve the total width of all elements at a position, or the width of a single element?
`padding`	Padding between elements at the same position. Elements are shrunk by this proportion to allow space between them. Defaults to 0.1.
`reverse`	If `TRUE`, will reverse the default stacking order. This is useful if you're rotating both the plot and legend.

Value

position methods

See Also

Other position adjustments: position_identityx(), position_points_sinax()

Examples

```
library(ggplot2)
library(patchwork)
iris$ID <- rep(c(rep("test1", 15), rep("test2", 15), rep("test3", 20)),3)
p <- ggplot(iris, aes(x=Species,y=Petal.Length,fill=ID))
p1 <- p + geom_bar(stat="identity",position=position_dodgex())
p2 <- p + geom_bar(stat="identity",position=position_dodgex(vexpand=5))
p3 <- ggplot(iris, aes(x=Petal.Length, y=Species, fill=ID)) +
      geom_bar(stat="identity", orientation="y",
               position=position_dodgex(hexpand=5))
p4 <- p1 + p2 + p3
p4
p5 <- p + geom_boxplot(position=position_dodgex2())
p6 <- p + geom_boxplot(position=position_dodgex2(vexpand=5))
p7 <- ggplot(iris, aes(x=Petal.Length, y=Species, fill=ID)) +
      geom_boxplot(orientation="y",
                   position=position_dodgex2(hexpand=5))
p8 <- p5 + p6 + p7
p8
```

---

position_identityx *adjust identity position which can be shifted vertically or horizontally.*

Description

adjust identity position which can be shifted vertically or horizontally.

Usage

```
position_identityx(hexpand = NA, vexpand = NA)
```

Arguments

hexpand	numeric, distance to be shifted horizontally for geoms that have a position, default is NA.
vexpand	numeric, distance to be shifted vertically for geoms that have a position, default is NA.

Value

position method.

Author(s)

Shuangbin Xu

See Also

Other position adjustments: position_dodgex(), position_points_sinax()

Examples

```
library(ggplot2)
library(patchwork)
p <- ggplot(mtcars, aes(x=wt, y=mpg))
p1 <- p + geom_point(position=position_identityx()) + ylim(0, 50)
# whole point layer was shifted vertically (distance=5).
# the label of axis y should be subtracted 5 to get the true value..
p2 <- p + geom_point(position=position_identityx(vexpand=5)) + ylim(0, 50)
# whole point layer was shifted horizontally (distance=5).
# the label of axis x should be subtracted 5 to get the true value.
p3 <- ggplot(mtcars, aes(y=wt, x=mpg)) +
      geom_point(position=position_identityx(hexpand=5)) + xlim(0, 50)
p4 <- p1 + p2 + p3
p4
```

---

position_jitterdodgex *Simultaneously dodge and jitter, and the whole layer can be shifted vertically or horizontally*

Description

This is primarily used for aligning points generated through 'geom_point()' with dodged boxplots (e.g., a 'geom_boxplot()' with a fill aesthetic supplied). And the points can be shifted vertically or horizontally with 'hexpand' or 'vexpand' arguments.

Usage

```
position_jitterdodgex(
  jitter.width = NULL,
  jitter.height = 0,
  dodge.width = 0.75,
  hexpand = NA,
  vexpand = NA,
  seed = NA
)
```

Arguments

jitter.width	degree of jitter in x direction. Defaults to 40% of the resolution of the data.
jitter.height	degree of jitter in y direction. Defaults to 0.
dodge.width	the amount to dodge in the x direction. Defaults to 0.75, the default position_dodge() width.
hexpand	The distance to be shifted vertically or horizontally, default is NA.

`vexpand` The distance to be shifted vertically or horizontally, default is NA.

`seed` A random seed to make the jitter reproducible. Useful if you need to apply the same jitter twice, e.g., for a point and a corresponding label. The random seed is reset after jittering. If `NA` (the default value), the seed is initialised with a random value; this makes sure that two subsequent calls start with a different seed. Use `NULL` to use the current random seed and also avoid resetting

---

`position_jitterx` *Jitter points to avoid overplotting, and the whole points can be shifted vertically or horizontally*

Description

This is the extension of 'position_jitter' of ggplot2, points are randomly shifted up and down and/or left and right. In addition, the whole points layer can be shifted by the 'hexpand' or 'vexpand' parameter. Counterintuitively adding random noise to a plot can sometimes make it easier to read. Jittering is particularly useful for small datasets with at least one discrete position.

Usage

```
position_jitterx(
  width = NULL,
  height = NULL,
  hexpand = NA,
  vexpand = NA,
  seed = NA
)
```

Arguments

`width, height` Amount of vertical and horizontal jitter. The jitter is added in both positive and negative directions, so the total spread is twice the value specified here.

If omitted, defaults to 40% of the resolution of the data: this means the jitter values will occupy 80% of the implied bins. Categorical data is aligned on the integers, so a width or height of 0.5 will spread the data so it's not possible to see the distinction between the categories.

`hexpand, vexpand` The distance to be shifted vertically or horizontally, default is NA.

`seed` A random seed to make the jitter reproducible. Useful if you need to apply the same jitter twice, e.g., for a point and a corresponding label. The random seed is reset after jittering. If `NA` (the default value), the seed is initialised with a random value; this makes sure that two subsequent calls start with a different seed. Use `NULL` to use the current random seed and also avoid resetting

Examples

```
library(ggtree)
library(treeio)
library(ggplot2)
set.seed(1024)
tr <- rtree(10)

df <- data.frame(id=tr$tip.label, group=rep(c("A", "B"),5))
dat <- data.frame(id=rep(tr$tip.label, 8), value=rnorm(80, 0.5, 0.15))
dt <- merge(dat, df, by.x="id", by.y="id")
p1 <- ggtree(tr) %<+% df +
      geom_tiplab(
          align=TRUE,
          linesize=.1,
          size=3
      )

gf1 <- geom_fruit(data=dat,
                  geom=geom_boxplot,
                  mapping=aes(x=value, y=id),
                  orientation="y",
                  offset=0.1,
                  pwidth=0.9

         )
set.seed(1024)
gf2 <- geom_fruit(
          data=dat,
          geom=geom_point,
          mapping=aes(x=value, y=id, color=group),
          offset=0.1,
          pwidth=0.9,
          position= position_jitterx(height=0.3),
          axis.params=list(axis="x", text.size=2),
          grid.params=list()
          )

p2 <- p1 + geom_fruit_list(gf1, gf2)
p2
```

`position_points_jitterx`

Randomly jitter the points in a ridgeline plot which can be shifted horizontally

---

Description

This is a position adjustment specifically for 'geom_density_ridges()' and related geoms. It only jitters the points drawn by these geoms, if any. If no points are present, the plot remains unchanged.

The effect is similar to position_jitter(): points are randomly shifted up and down and/or left and right. It add 'hexpand' that can control shift horizontally.

Usage

```
position_points_jitterx(
  width = 0,
  height = 0.2,
  yoffset = 0,
  hexpand = NA,
  adjust_vlines = FALSE,
  seed = NULL
)
```

Arguments

width	Width for horizontal jittering. By default set to 0.
height	Height for vertical jittering, applied in both directions (up and down). By default 0.2.
yoffset	Vertical offset applied in addition to jittering.
hexpand	numeric, distance to be shifted horizontally for geoms that have a position, default is NA.
adjust_vlines	If TRUE, adjusts vertical lines (as are drawn for quantile lines, for example) to align with the point cloud.
seed	Random seed. If set to NULL, the current random number generator is used. If set to NA, a new random random seed is generated. If set to a number, this number is used as seed for jittering only.

See Also

Other position adjustments for ridgeline plots: position_points_sinax, position_raincloudx

---

position_points_sinax *adjust ridgeline plot position which can be shifted vertically or horizontally.*

Description

This is a position adjustment specifically for 'geom_density_ridges()', but it add 'hexpand' that can control shift horizontally.

Usage

```
position_points_sinax(
  rel_min = 0.02,
  rel_max = 0.98,
  seed = NULL,
  hexpand = NA
)
```

Arguments

rel_min	numeric, the relative minimum value at which a point can be placed.
rel_max	numeric, the relative maximum value at which a point can be placed.
seed	numeric, Random seed, if set to NULL, the current random number generator is used. If set to NA, a new random random seed is generated. If set to a number, this number is used as seed for jittering only, default is NULL.
hexpand	numeric, distance to be shifted horizontally for geoms that have a position, default is NA.

Value

position method.

Author(s)

Shuangbin Xu

See Also

Other position adjustments: position_dodgex(), position_identityx()

---

position_raincloudx — *Create a cloud of randomly jittered points below a ridgeline plot which can be shifted horizontally*

Description

This is a position adjustment specifically for 'geom_density_ridges()' and related geoms. It only jitters the points drawn by these geoms, if any. If no points are present, the plot remains unchanged. The effect is similar to position_points_jitterx(), only that by default the points lie all underneath the baseline of each individual ridgeline. It add 'hexpand' that can control shift horizontally.

Usage

```
position_raincloudx(
  width = 0,
  height = 0.4,
  ygap = 0.05,
  hexpand = NA,
  adjust_vlines = FALSE,
  seed = NULL
)
```

Arguments

width	Width for horizontal jittering. By default set to 0.
height	Total height of point cloud. By default 0.4.
ygap	Vertical gap between ridgeline baseline and point cloud.
hexpand	numeric, distance to be shifted horizontally for geoms that have a position, default is NA.
adjust_vlines	If TRUE, adjusts vertical lines (as are drawn for quantile lines, for example) to align with the point cloud.
seed	Random seed. See position_points_jitterx.

Details

The idea for this position adjustment comes from Micah Allen, who proposed this type of plot in a blog post on March 15, 2018.

See Also

Other position adjustments for ridgeline plots: position_points_jitterx, position_points_sinax

---

position_stackx — *Stack overlapping objects which can be shifted vertically or horizontally*

Description

Stack overlapping objects which can be shifted vertically or horizontally

Usage

```
position_stackx(vjust = 1, hexpand = NA, vexpand = NA, reverse = FALSE)
```

Arguments

vjust	Vertical adjustment for geoms that have a position (like points or lines), not a dimension (like bars or areas). Set to 0 to align with the bottom, 0.5 for the middle, and 1 (the default) for the top.
hexpand	numeric, distance to be shifted horizontally for geoms that have a position, default is NA.
vexpand	numeric, distance to be shifted vertically for geoms that have a position, default is NA.
reverse	If TRUE, will reverse the default stacking order. This is useful if you're rotating both the plot and legend.

Value

position method.

Author(s)

Shuangbin Xu

Examples

```
library(ggplot2)
library(patchwork)
df <- data.frame(trt = c("a", "b", "c"), outcome = c(2.3, 1.9, 3.2))
#
p1 <- ggplot(df, aes(x=trt,  y=outcome)) +
      geom_bar(stat="identity",
               position=position_stackx())

p2 <- ggplot(df, aes(x=trt, y=outcome)) +
      geom_bar(stat="identity",
               position=position_stackx(vexpand=5))

p3 <- ggplot(df, aes(x=outcome, y=trt)) +
      geom_bar(stat="identity",
               orientation="y",
               position=position_stackx(hexpand=5))
p <- p1 + p2 + p3
p
```

Index

∗ **datasets**
PositionDodgex, 8
PositionJitterdodgex, 8
PositionJitterx, 8
∗ **position adjustments**
position_dodgex, 9
position_identityx, 10
position_points_sinax, 14

fruit_plot *(*geom_fruit*)*, 2

geom_fruit, 2
geom_fruit_list, 6

position_dodgex, 9, *11*, *15*
position_dodgex2 *(*position_dodgex*)*, 9
position_identityx, *10*, 10, *15*
position_jitter(), *14*
position_jitterdodgex, 11
position_jitterx, 12
position_points_jitterx, 13, *16*
position_points_jitterx(), *15*
position_points_sinax, *10, 11*, *14*, 14, *16*
position_raincloudx, *14*, 15
position_stackx, 16
PositionDodgex, 8
PositionDodgex2 *(*PositionDodgex*)*, 8
PositionIdentityx *(*PositionDodgex*)*, 8
PositionJitterdodgex, 8
PositionJitterx, 8
PositionPointsJitterx *(*PositionDodgex*)*, 8
PositionPointsSinax *(*PositionDodgex*)*, 8
PositionRaincloudx *(*PositionDodgex*)*, 8
PositionStackx *(*PositionDodgex*)*, 8