Package: phylepic (via r-universe)

July 5, 2024

Title Combined Visualisation of Phylogenetic and Epidemiological Data

Version 0.2.1.9000

Description A collection of utilities and 'ggplot2' extensions to assist with visualisations in genomic epidemiology. This includes the 'phylepic' chart, a visual combination of a phylogenetic tree and a matched epidemic curve. The included 'ggplot2' extensions such as date axes binned by week are relevant for other applications in epidemiology and beyond. The approach is described in Suster et al. (2024) <doi:10.1101/2024.04.02.24305229>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Imports ape, cli, cowplot, dplyr, forcats, ggnewscale, ggplot2 (>= 3.5.0), ggraph, igraph, rlang, scales, tidygraph, vctrs

URL https://github.com/cidm-ph/phylepic, https://cidm-ph.github.io/phylepic/

BugReports https://github.com/cidm-ph/phylepic/issues

Config/testthat/edition 3

Repository https://cidm-ph.r-universe.dev

RemoteUrl https://github.com/cidm-ph/phylepic

RemoteRef HEAD

RemoteSha 8cdf6655462e60f94576fca2e588aaed103fa91e

Contents

`coord_tree` *Cartesian coordinates with specialised grid for trees*

Description

This coord is based on the default Cartesian coordinates, but draws the a filled background in addition to the normal grid lines. The grid is forced to appear on every integer value within the scale's range.

Usage

```
coord_tree(
  xlim = NULL,
  ylim = NULL,
  expand = TRUE,
  default = FALSE,
  clip = "on"
)
```

Arguments

`xlim`, `ylim`, `expand`, `default`, `clip`
See `ggplot2::coord_cartesian()`

Details

The appearance of the grid can be controlled with theme elements:

`phylepic.grid.bar` filled grid (`element_rect()`).

`phylepic.grid.line` grid line (`element_line()`).

`phylepic.grid.every` grid frequency (`integer`). Default for both `phylepic.grid.every.bar` and `phylepic.grid.every.stripe`

`phylepic.grid.every.bar` grid bar frequency (`integer`). Defaults to 2 to give an alternative striped background

`phylepic.grid.every.stripe` grid bar frequency (`integer`). Defaults to 1 so that every tip on a tree has its own line

Value

coord suitable for adding to a plot

`create_tree_layout` *Create a graph layout for plotting*

Description

This lays out a graph using `ggraph::create_layout()` with the `"dendrogram"` layout, takes edge lengths from the tree, and flips the layout coordinates. The plotting functions associated with `phylepic()` expect the graph to be laid out using these settings.

Usage

```
create_tree_layout(tree, tip_data = NULL)
```

Arguments

`tree`	A tree-like graph or a `phylepic` object.
`tip_data`	A data frame with tip metadata. There must be a column called `.phylepic.name` with values that correspond to the names of leaf nodes in the tree. If `NULL`, no tip data is joined onto the tree.

Value

A "layout_ggraph" object suitable for plotting with ggplot2::ggplot‘.

drop.clade *Drop a clade from a phylogentic tree*

Description

drop.clade invokes ape::drop.tip() on all tips descendent from the specified node. This is convenient when used alongside ape::getMRCA() to drop a clade defined by the most recent common ancestor of a set of tips, rather than exhaustively specifying all of its tips.

Usage

```
drop.clade(phy, node, root.edge = 0, collapse.singles = TRUE)
```

Arguments

phy	an object of class "phylo".
node	number specifying the parent node of the clade to delete.
root.edge, collapse.singles	passed to ape::drop.tip().

Value

New phylo object with the chosen clade removed

Examples

```
library("ape")
data(bird.orders)
plot(bird.orders)

# find the common ancestor of some tips
mrca <- ape::getMRCA(bird.orders, c("Passeriformes", "Coliiformes"))

# drop the clade descending from that ancestor
plot(drop.clade(bird.orders, mrca))
```

GeomCalendar *Specialised tile geometry for calendar plots*

Description

This geom behaves mostly the same as ggplot2::geom_tile() with a few additions. Firstly, the label aesthetic is supported to draw text on top of the tiles. Secondly, out of bounds values can be drawn as arrows at the edge of the scale (see details below).

Usage

```
geom_calendar(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  linejoin = "mitre",
  label_params = list(colour = "grey30"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping, data, stat, position, linejoin, na.rm, show.legend, inherit.aes, ...	see ggplot2::geom_tile().
label_params	additional parameters for text labels if present (see ggplot2::geom_text()).

Details

Any x values that are infinite (i.e. -Inf or Inf) would normally be dropped by ggplot's layers. If any such values survive the stat processing, they will be drawn by geom_calendar() as triangles at the respective edges of the scale. This is intended to work with a scale configured to use oob_infinite() for out of bounds handling. The triangles are drawn with their base (vertical edge) sitting on the scale limit, and their width equal to half of the median bin width.

Note that the label aesthetic will be dropped if the data are not grouped in the expected way. In general this means that all rows contributing to a given bin must have the same value for the label aesthetic.

Examples

```
library(ggplot2)

set.seed(1)
events <- rep(as.Date("2024-01-31") - 0:30, rpois(31, 6))
values <- round(rgamma(length(events), 1, 0.01))
df <- data.frame(date = events, value = values)

ggplot(df) +
    geom_calendar(
        aes(date, value, label = after_stat(count)),
        colour = "white",
        stat = "week_2d",
        week_start = "Monday",
        bins.y = 10
    ) +
```

```
    scale_x_week(
        limits = as.Date(c("2024-01-08", NA)),
        expand = expansion(add = 3.5)
    )
```

`geom_node_text_filled` *Annotate nodes with text and a background*

Description

This geom behaves like `ggraph::geom_node_text()` except that it also inserts a white background behind the text extending to the left margin. This will only make sense for a horizontal dendrogram graph layout with the root node on the left.

Usage

```
geom_node_text_filled(
  mapping = NULL,
  data = NULL,
  position = "identity",
  parse = FALSE,
  check_overlap = FALSE,
  show.legend = NA,
  ...
)
```

Arguments

`mapping`, `data`, `position`, `parse`, `check_overlap`, `show.legend`, `...`
: Arguments passed to the geom that powers `ggraph::geom_node_text()`. Note that the additional arguments of that function such as `repel` are not supported here.

Details

This background covers up part of the grid rendered by the coord layer. The reason that this is done as part of the text instead of as a separate layer is so that we have access to the rendered dimensions of the text grobs.

Value

Layer that draws text and background grobs

oob_infinite *Out of bounds handling*

Description

This helper works the same way as `scales::oob_censor()` and similar. Out of bounds values are pushed to positive or negative infinity. This is not useful for builtin ggplot layers which will display a warning and drop rows with infinite values in required aesthetics. `geom_calendar()` however uses the infinite values to indicate out of bounds values explicitly on the plot.

Usage

```
oob_infinite(x, range = c(0, 1))
```

Arguments

`x`	A numeric vector of values to modify.
`range`	A numeric vector of length two giving the minimum and maximum limit of the desired output range respectively.

Value

A numerical vector of the same length as `x` where out of bound values have been replaced by `Inf` or `-Inf` accordingly.

phylepic *Combine metadata (a line list) with a phylogenetic tree*

Description

Some checks are performed to catch issues where the metadata and tree tips don't match up. Any columns in `metadata` that are factors have all levels that do not appear in the data dropped.

Usage

```
phylepic(
  tree,
  metadata,
  name,
  date,
  unmatched_tips = c("error", "drop", "keep")
)
```

Arguments

tree	An object convertible to a tbl_graph. This will usually be a "phylo" object, but see tidygraph::tbl_graph for more details.
metadata	A data frame.
name	Column in metadata that corresponds to the tree's tip labels (tidy-eval).
date	Column in metadata that contains the date data (class "Date") for the tips (tidy-eval).
unmatched_tips	Action to take when tree contains tip labels that do not appear in name. "error" aborts with an error message, "drop" drops unmatched tips from tree, "keep".

Details

To reduce surprises when matching metadata and tree, by default an error occurs when there are tree tips that do not have associated metadata. On the other hand, it it expected that metadata might contain rows that do not correspond to the tips in tree.

This often means that factor columns from metadata will contain levels that do not appear at all in the tree. For plotting, ggplot2::discrete_scale normally solves this with drop = TRUE, however this can lead to inconsistencies when sharing the same scale across multiple phylepic panels. phylepic() drops unused levels in all factors so that scales can use drop = FALSE for consistency.

Value

An object of class "phylepic".

Examples

```
library(ape)

tree <- read.tree(system.file("enteric.newick", package = "phylepic"))
metadata <- read.csv(
  system.file("enteric_metadata.csv", package = "phylepic")
)
phylepic(tree, metadata, name, as.Date(collection_date))
```

---

plot.phylepic *Plot "phylepic" objects*

Description

The autoplot() and plot() methods for "phylepic" objects assemble various panels into the final plot. To facilitate customisations, the plots from each panel can be overwritten. Some effort is made to ensure that the specified plots will look reasonable when assembled.

Usage

```
## S3 method for class 'phylepic'
plot(
  x,
  ...,
  plot.tree = plot_tree(),
  plot.bars = plot_bars(),
  plot.calendar = plot_calendar(),
  plot.epicurve = plot_epicurve(),
  scale.date = NULL,
  scale.fill = NULL,
  width.tree = 10,
  width.bars = 1,
  width.date = 5,
  width.legend = 2,
  height.tree = 2
)

## S3 method for class 'phylepic'
autoplot(
  object,
  ...,
  plot.tree = plot_tree(),
  plot.bars = plot_bars(),
  plot.calendar = plot_calendar(),
  plot.epicurve = plot_epicurve(),
  scale.date = NULL,
  scale.fill = NULL,
  width.tree = 10,
  width.bars = 1,
  width.date = 5,
  width.legend = 2,
  height.tree = 2
)
```

Arguments

`...`	Ignored.
`plot.tree`	ggplot for the tree panel (see plot_tree).
`plot.bars`	ggplot for the metadata bars panel (see plot_bars).
`plot.calendar`	ggplot for the calendar panel (see plot_calendar).
`plot.epicurve`	ggplot for the epidemic curve panel (see plot_epicurve).
`scale.date`	A date scale passed to both the calendar and epicurve panels (see ggplot2::scale_x_date).
`scale.fill`	A fill scale passed to both the calendar and epicurve panels (see ggplot2::scale_x_date).
`width.tree`	Relative width of the tree panel.
`width.bars`	Relative width of the metadata bars panel.

width.date	Relative width of the calendar panel.
width.legend	Relative width of the legend, if present.
height.tree	Relative height of the tree panel.
object, x	Object of class "phylepic".

Details

In general, if you wish to suppress a panel from the plot, set the corresponding plot.* argument to NULL. To customise it, use the corresponding plot_*() function, which returns a ggplot plot. You can then add new layers or themes to that plot. See vignette("phylepic") for examples.

Legends from all panels are collected and de-duplicated. They are drawn on the right edge of the overall plot.

Value

plot() is usually called to display the plot, whereas autoplot() returns a "ggplot" object that can later be displayed with print().

See Also

Other phylepic plots: plot_bars(), plot_calendar(), plot_epicurve(), plot_tree()

---

plot_bars *Plot metadata bars panel*

Description

This uses ggplot2::geom_tile() to produce a grid with a row aligned with each tip on the tree, and a column for each type of data specified. If no scales are specified, one is created for each factor column in the metadata table.

Usage

```
plot_bars(phylepic, ...)
```

Arguments

phylepic	object of class "phylepic".
...	scale specifications.

Value

If phylepic is specified returns a ggplot, otherwise a function that when passed a "phylepic" object produces a ggplot for use with plot.phylepic().

See Also

Other phylepic plots: plot.phylepic(), plot_calendar(), plot_epicurve(), plot_tree()

plot_calendar *Plot calendar panel*

Description

Plot calendar panel

Usage

```
plot_calendar(
  phylepic,
  fill = NULL,
  weeks = TRUE,
  week_start = getOption("phylepic.week_start"),
  labels = NULL,
  labels.params = list(size = 3, fontface = "bold", colour = "white")
)
```

Arguments

phylepic	Object of class "phylepic".
fill	Variable in metadata table to use for the fill aesthetic (tidy-eval).
weeks	When TRUE, bin the date axis by weeks.
week_start	Day the week begins (defaults to Monday). Can be specified as a case-insensitive English weekday name such as "Monday" or an integer. Since you generally won't want to mix definitions, it is more convenient to control this globally with the "phylepic.week_start" option, e.g. options(phylepic.week_start = "Monday").
labels	Controls the format of date labels on calendar tiles. If NULL, no labels are drawn. If a character scalar, controls the date format (see strptime()).
labels.params	Passed to ggplot2::geom_text() if labels are drawn.

Value

If phylepic is specified returns a ggplot, otherwise a function that when passed a "phylepic" object produces a ggplot for use with plot.phylepic().

See Also

Other phylepic plots: plot.phylepic(), plot_bars(), plot_epicurve(), plot_tree()

plot_epicurve *Plot epidemic curve panel*

Description

Plot epidemic curve panel

Usage

```
plot_epicurve(
  phylepic,
  fill = NULL,
  weeks = TRUE,
  week_start = getOption("phylepic.week_start")
)
```

Arguments

phylepic	Object of class "phylepic".
fill	Variable in metadata table to use for the fill aesthetic (tidy-eval).
weeks	When TRUE, bin the date axis by weeks.
week_start	Day the week begins (defaults to Monday). Can be specified as a case-insensitive English weekday name such as "Monday" or an integer. Since you generally won't want to mix definitions, it is more convenient to control this globally with the "phylepic.week_start" option, e.g. options(phylepic.week_start = "Monday").

Value

If phylepic is specified returns a ggplot, otherwise a function that when passed a "phylepic" object produces a ggplot for use with plot.phylepic().

See Also

Other phylepic plots: plot.phylepic(), plot_bars(), plot_calendar(), plot_tree()

plot_tree *Plot phylogenetic tree panel*

Description

The tree is drawn using ggraph with its dendrogram layout. When customising it, you may wish to add layers such as ggraph::geom_node_point(). The metadata table is joined onto the tree, so all its column names are available for use in the various ggraph geoms.

Usage

```
plot_tree(phylepic, label = .data$name, bootstrap = TRUE)
```

Arguments

phylepic	object of class "phylepic".
label	variable in metadata table corresponding to the tip labels (tidy-eval).
bootstrap	when TRUE, draw bootstrap vaues on the tree. These are only drawn if they are detected from the node labels having the form "a/b" where both "a" and "b" are numbers. Currently, the bootstrap values are displayed as a percentage, suppressing zero values and values for very short branches. To customise the appearance or details instead use bootstrap = FALSE and add your own layer with ggraph::geom_edge_elbow.

Value

If phylepic is specified returns a ggplot, otherwise a function that when passed a "phylepic" object produces a ggplot for use with plot.phylepic().

See Also

Other phylepic plots: plot.phylepic(), plot_bars(), plot_calendar(), plot_epicurve()

---

scale_x_week *Date scale with breaks specified by week*

Description

This produces a scale that is measured in days as with ggplot2::scale_x_date, however it will snap breaks and limits to week boundaries so that things work as intended when binning by week.

Usage

```
scale_x_week(
  name = waiver(),
  week_breaks = waiver(),
  labels = waiver(),
  date_labels = waiver(),
  week_minor_breaks = waiver(),
  oob = oob_infinite,
  limits = NULL,
  ...,
  week_start = getOption("phylepic.week_start")
)
```

Arguments

`name`, `labels`, `date_labels`, `oob`, `limits`, `...`
: see `ggplot2::scale_x_date()`.

`week_breaks`, `week_minor_breaks`
: frequency of breaks in number of weeks (e.g. `2` for fortnightly breaks).

`week_start`
: Day the week begins (defaults to Monday). Can be specified as a case-insensitive English weekday name such as "Monday" or an integer. Since you generally won't want to mix definitions, it is more convenient to control this globally with the `"phylepic.week_start"` option, e.g. `options(phylepic.week_start = "Monday")`.

Details

Any `limits` specified are converted to the nearest week boundary that includes the specified dates, i.e. the lower limit will be rounded down and the upper limit rounded up so that the limits are week boundaries.

Value

a ggplot scale object.

`stat_week` *Calculate week bins from dates*

Description

Computes weeks for date data. This is mostly equivalent to `ggplot2::stat_bin()` with the bins fixed to weeks starting on a particular day.

Usage

```
stat_week(
  mapping = NULL,
  data = NULL,
  geom = "bar",
  position = "stack",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  week_start = getOption("phylepic.week_start"),
  pad = FALSE
)
```

Arguments

`mapping`, `data`, `geom`, `position`, `na.rm`, `show.legend`, `inherit.aes`, `pad`, `...`
: See `ggplot2::stat_bin()`.

`week_start`
: Day the week begins (defaults to Monday). Can be specified as a case-insensitive English weekday name such as "Monday" or an integer. Since you generally won't want to mix definitions, it is more convenient to control this globally with the `"phylepic.week_start"` option, e.g. `options(phylepic.week_start = "Monday")`.

Value

ggplot2 stat layer.

Examples

```
library(ggplot2)

set.seed(1)
events <- rep(as.Date("2024-01-31") - 0:30, rpois(31, 2))
df <- data.frame(date = events)

ggplot(df) + stat_week(aes(date), week_start = "Monday")

# or equivalently:
# ggplot(df) + geom_bar(aes(date), stat = "week", week_start = "Monday")
```

---

stat_week_2d *Calculate week bins with additional binning in the y axis*

Description

Computes week bins for date data in the x aesthetic, and allows the binning to be specified for the y aesthetic. This is mostly equivalent to `ggplot2::stat_bin_2d()` with the x aesthetic handling fixed to weeks.

Usage

```
stat_week_2d(
  mapping = NULL,
  data = NULL,
  geom = "tile",
  position = "identity",
  ...,
  bins.y = NULL,
  binwidth.y = NULL,
  breaks.y = NULL,
  center.y = NULL,
```

```
  boundary.y = NULL,
  closed.y = c("left", "right"),
  drop = TRUE,
  week_start = getOption("phylepic.week_start"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`, `data`, `geom`, `position`, `na.rm`, `show.legend`, `inherit.aes`, `...`	See ggplot2::stat_bin_2d.
`bins.y`, `binwidth.y`, `breaks.y`, `center.y`, `boundary.y`, `closed.y`	See the analogous parameters in ggplot2::stat_bin_2d.
`drop`	drop bins with zero count.
`week_start`	Day the week begins (defaults to Monday). Can be specified as a case-insensitive English weekday name such as "Monday" or an integer. Since you generally won't want to mix definitions, it is more convenient to control this globally with the `"phylepic.week_start"` option, e.g. `options(phylepic.week_start = "Monday")`.

Details

The computed aesthetics are similar to those of `stat_bin_2d()`, including `after_stat(count)`, `after_stat(density)`, and the bin positions and sizes: `after_stat(xmin)`, `after_stat(height)`, and so on.

Value

ggplot2 stat layer.

Examples

```
library(ggplot2)

set.seed(1)
events <- rep(as.Date("2024-01-31") - 0:30, rpois(31, 6))
values <- round(rgamma(length(events), 1, 0.01))
df <- data.frame(date = events, value = values)

ggplot(df) + stat_week_2d(aes(date, value), week_start = "Monday")
```

`week_breaks` *Breaks for week-binning date axes*

Description

Breaks for week-binning date axes

Usage

```
week_breaks(width = 1L, week_start = getOption("phylepic.week_start"))
```

Arguments

`width`	Number of weeks between breaks (e.g. 2 will give a break every fortnight).
`week_start`	Day the week begins (defaults to Monday). Can be specified as a case-insensitive English weekday name such as "Monday" or an integer. Since you generally won't want to mix definitions, it is more convenient to control this globally with the `"phylepic.week_start"` option, e.g. `options(phylepic.week_start = "Monday")`.

Value

A break function suitable for use in `ggplot2::scale_x_date()` et al.

Index