Package 'vistime'

October 29, 2023

Title Pretty Timelines in R

Version 1.2.4

Date 2023-10-28

Description A library for creating time based charts, like Gantt or timelines. Possible outputs include 'ggplot2' diagrams, 'plotly.js' graphs, 'Highcharts.js' widgets and data.frames. Results can be
used in the 'RStudio' viewer pane, in 'RMarkdown' documents or in Shiny apps. In the interactive outputs created by vistime() and hc_vistime(), you can interact with the plot using mouse hover or zoom.

License GPL-3 | file LICENSE

URL https://shosaco.github.io/vistime/

BugReports https://github.com/shosaco/vistime/issues

Depends R (>= 3.2.0)

Imports rlang, assertthat (>= 0.1), plotly (>= 4.0.0), ggplot2 (>= 3.4.0), ggrepel (>= 0.7.0), RColorBrewer (>= 0.2.2)

Encoding UTF-8

RoxygenNote 7.2.3

Suggests prettydoc, knitr, rmarkdown, testthat (>= 3.0.0), covr, highcharter (> 0.1.0)

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Sandro Raabe [aut, cre]

Maintainer Sandro Raabe <sa.ra.online@posteo.de>

Repository CRAN

Date/Publication 2023-10-29 21:30:02 UTC

R topics documented:

`vistime-package` *vistime: Pretty Timeline Charts in R*

Description

A library for creating time based charts, like Gantt or timelines. Possible outputs include ggplot2 diagrams, plotly.js graphs, Highcharts.js widgets and data.frames. Results can be used in the RStudio viewer pane, in RMarkdown documents or in Shiny apps. In the interactive outputs created by vistime() and hc_vistime(), you can interact with the plot using mouse hover or zoom.

Details

Pretty timelines in R

Author(s)

Sandro Raabe <sa.ra.online@posteo.de>

See Also

Useful links:

- https://shosaco.github.io/vistime/
- Report bugs at https://github.com/shosaco/vistime/issues

`gg_vistime` *Create a Timeline rendered by ggplot2*

Description

Provide a data frame with event data to create a static timeline plot created by ggplot2. Simplest drawable dataframe can have columns ‘event‘ and ‘start‘.

Usage

```
gg_vistime(
  data,
  col.event = "event",
  col.start = "start",
  col.end = "end",
  col.group = "group",
  col.color = "color",
  col.fontcolor = "fontcolor",
  optimize_y = TRUE,
  linewidth = NULL,
  title = NULL,
  show_labels = TRUE,
  background_lines = NULL,
  ...
)
```

Arguments

data	data.frame that contains the data to be visualized
col.event	(optional, character) the column name in data that contains event names. Default: *event*.
col.start	(optional, character) the column name in data that contains start dates. Default: *start*.
col.end	(optional, character) the column name in data that contains end dates. Default: *end*.
col.group	(optional, character) the column name in data to be used for grouping. Default: *group*.
col.color	(optional, character) the column name in data that contains colors for events. Default: *color*, if not present, colors are chosen via RColorBrewer.
col.fontcolor	(optional, character) the column name in data that contains the font color for event labels. Default: *fontcolor*, if not present, color will be black.
optimize_y	(optional, logical) distribute events on y-axis by smart heuristic (default), otherwise use order of input data.
linewidth	(optional, numeric) the linewidth (in pixel) for the events (typically used for large amount of parallel events). Default: heuristic value.
title	(optional, character) the title to be shown on top of the timeline. Default: NULL.
show_labels	(optional, boolean) choose whether or not event labels shall be visible. Default: TRUE.
background_lines	(optional, integer) the number of vertical lines to draw in the background to demonstrate structure (default: heuristic).
...	for deprecated arguments up to vistime 1.1.0 (like events, colors, ...)

Value

gg_vistime returns an object of class gg and ggplot.

See Also

Functions ?vistime and ?hc_vistime for different charting engines (Plotly and Highcharts).

Examples

```
# presidents and vice presidents
pres <- data.frame(
  Position = rep(c("President", "Vice"), each = 3),
  Name = c("Washington", rep(c("Adams", "Jefferson"), 2), "Burr"),
  start = c("1789-03-29", "1797-02-03", "1801-02-03"),
  end = c("1797-02-03", "1801-02-03", "1809-02-03"),
  color = c("#cbb69d", "#603913", "#c69c6e")
)

gg_vistime(pres, col.event = "Position", col.group = "Name", title = "Presidents of the USA")

## Not run:
# ------ It is possible to change all attributes of the timeline using ggplot2::theme()
data <- read.csv(text="event,start,end
                       Phase 1,2020-12-15,2020-12-24
                       Phase 2,2020-12-23,2020-12-29
                       Phase 3,2020-12-28,2021-01-06
                       Phase 4,2021-01-06,2021-02-02")

p <- gg_vistime(data, optimize_y = T, col.group = "event", title = "ggplot customization example")

library(ggplot2)
p + theme(
  plot.title = element_text(hjust = 0, size=30),
  axis.text.x = element_text(size = 30, color = "violet"),
  axis.text.y = element_text(size = 30, color = "red", angle = 30),
  panel.border = element_rect(linetype = "dashed", fill=NA),
  panel.background = element_rect(fill = 'green')) +
  coord_cartesian(ylim = c(0.7, 3.5))

## End(Not run)
```

---

hc_vistime *Create a Timeline rendered by Highcharts.js*

Description

Provide a data frame with event data to create a visual and interactive timeline plot rendered by Highcharts. Simplest drawable dataframe can have columns ‘event‘ and ‘start‘. This feature is facilitated by the ‘highcharter‘ package, so, this package needs to be installed before attempting to produce any ‘hc_vistime()‘ output.

Usage

```
hc_vistime(
  data,
  col.event = "event",
  col.start = "start",
  col.end = "end",
  col.group = "group",
  col.color = "color",
  col.tooltip = "tooltip",
  optimize_y = TRUE,
  title = NULL,
  show_labels = TRUE,
  ...
)
```

Arguments

data	data.frame that contains the data to be visualized
col.event	(optional, character) the column name in data that contains event names. Default: *event*.
col.start	(optional, character) the column name in data that contains start dates. Default: *start*.
col.end	(optional, character) the column name in data that contains end dates. Default: *end*.
col.group	(optional, character) the column name in data to be used for grouping. Default: *group*.
col.color	(optional, character) the column name in data that contains colors for events. Default: *color*, if not present, colors are chosen via RColorBrewer.
col.tooltip	(optional, character) the column name in data that contains the mouseover tooltips for the events. Default: *tooltip*, if not present, then tooltips are built from event name and date.
optimize_y	(optional, logical) distribute events on y-axis by smart heuristic (default), otherwise use order of input data.
title	(optional, character) the title to be shown on top of the timeline. Default: NULL.
show_labels	(optional, boolean) choose whether or not event labels shall be visible. Default: TRUE.
...	for deprecated arguments up to vistime 1.1.0 (like events, colors, ...)

Value

hc_vistime returns an object of class highchart and htmlwiget

See Also

Functions ?vistime and ?gg_vistime for different charting engines (Plotly and ggplot2).

Examples

```
# presidents and vice presidents
pres <- data.frame(
  Position = rep(c("President", "Vice"), each = 3),
  Name = c("Washington", rep(c("Adams", "Jefferson"), 2), "Burr"),
  start = c("1789-03-29", "1797-02-03", "1801-02-03"),
  end = c("1797-02-03", "1801-02-03", "1809-02-03"),
  color = c("#cbb69d", "#603913", "#c69c6e")
)

hc_vistime(pres, col.event = "Position", col.group = "Name", title = "Presidents of the USA")
#'
## Not run:
# ------ It is possible to change all attributes of the timeline using highcharter::hc_*():
data <- read.csv(text="event,start,end
                       Phase 1,2020-12-15,2020-12-24
                       Phase 2,2020-12-23,2020-12-29
                       Phase 3,2020-12-28,2021-01-06
                       Phase 4,2021-01-06,2021-02-02")

library(highcharter)
p <- hc_vistime(data, optimize_y = T, col.group = "event",
                title = "Highcharts customization example")
p %>% hc_title(style = list(fontSize=30)) %>%
      hc_yAxis(labels = list(style = list(fontSize=30, color="violet"))) %>%
      hc_xAxis(labels = list(style = list(fontSize=30, color="red"), rotation=30)) %>%
      hc_chart(backgroundColor = "lightgreen")

## End(Not run)
```

`vistime` *Create a Timeline rendered by Plotly*

Description

Provide a data frame with event data to create a visual and interactive timeline plot rendered by Plotly. Simplest drawable dataframe can have columns ‘event‘ and ‘start‘.

Usage

```
vistime(
  data,
  col.event = "event",
  col.start = "start",
  col.end = "end",
  col.group = "group",
  col.color = "color",
  col.fontcolor = "fontcolor",
  col.tooltip = "tooltip",
```

```
  optimize_y = TRUE,
  linewidth = NULL,
  title = NULL,
  show_labels = TRUE,
  background_lines = NULL,
  ...
)
```

Arguments

data	data.frame that contains the data to be visualized
col.event	(optional, character) the column name in data that contains event names. Default: *event*.
col.start	(optional, character) the column name in data that contains start dates. Default: *start*.
col.end	(optional, character) the column name in data that contains end dates. Default: *end*.
col.group	(optional, character) the column name in data to be used for grouping. Default: *group*.
col.color	(optional, character) the column name in data that contains colors for events. Default: *color*, if not present, colors are chosen via RColorBrewer.
col.fontcolor	(optional, character) the column name in data that contains the font color for event labels. Default: *fontcolor*, if not present, color will be black.
col.tooltip	(optional, character) the column name in data that contains the mouseover tooltips for the events. Default: *tooltip*, if not present, then tooltips are built from event name and date.
optimize_y	(optional, logical) distribute events on y-axis by smart heuristic (default), otherwise use order of input data.
linewidth	(optional, numeric) the linewidth (in pixel) for the events (typically used for large amount of parallel events). Default: heuristic value.
title	(optional, character) the title to be shown on top of the timeline. Default: NULL.
show_labels	(optional, boolean) choose whether or not event labels shall be visible. Default: TRUE.
background_lines	(optional, integer) the number of vertical lines to draw in the background to demonstrate structure (default: 10). Less means more memory-efficient plot.
...	for deprecated arguments up to vistime 1.1.0 (like events, colors, ...)

Value

vistime returns an object of class plotly and htmlwidget.

See Also

Functions ?hc_vistime and ?gg_vistime for different charting engines (Highcharts and ggplot2).

Examples

```
# presidents and vice presidents
pres <- data.frame(
  Position = rep(c("President", "Vice"), each = 3),
  Name = c("Washington", rep(c("Adams", "Jefferson"), 2), "Burr"),
  start = c("1789-03-29", "1797-02-03", "1801-02-03"),
  end = c("1797-02-03", "1801-02-03", "1809-02-03"),
  color = c("#cbb69d", "#603913", "#c69c6e"),
  fontcolor = c("black", "white", "black")
)

vistime(pres, col.event = "Position", col.group = "Name", title = "Presidents of the USA")


## Not run:
# Argument`optimize_y` can be used to change the look of the timeline. `TRUE` (the default)
# will find a nice heuristic to save `y`-space, distributing the events:
data <- read.csv(text="event,start,end
                       Phase 1,2020-12-15,2020-12-24
                       Phase 2,2020-12-23,2020-12-29
                       Phase 3,2020-12-28,2021-01-06
                       Phase 4,2021-01-06,2021-02-02")

vistime(data, optimize_y = TRUE)

# `FALSE` will plot events as-is, not saving any space:
vistime(data, optimize_y = FALSE)


# more complex and colorful example
data <- read.csv(text = "event,group,start,end,color
Phase 1,Project,2018-12-22,2018-12-23,#c8e6c9
Phase 2,Project,2018-12-23,2018-12-29,#a5d6a7
Phase 3,Project,2018-12-29,2019-01-06,#fb8c00
Phase 4,Project,2019-01-06,2019-02-02,#DD4B39
Room 334,Team 1,2018-12-22,2018-12-28,#DEEBF7
Room 335,Team 1,2018-12-28,2019-01-05,#C6DBEF
Room 335,Team 1,2019-01-05,2019-01-23,#9ECAE1
Group 1,Team 2,2018-12-22,2018-12-28,#E5F5E0
Group 2,Team 2,2018-12-28,2019-01-23,#C7E9C0
3-200,category 1,2018-12-25,2018-12-25,#1565c0
3-330,category 1,2018-12-25,2018-12-25,#1565c0
3-223,category 1,2018-12-28,2018-12-28,#1565c0
3-225,category 1,2018-12-28,2018-12-28,#1565c0
3-226,category 1,2018-12-28,2018-12-28,#1565c0
3-226,category 1,2019-01-19,2019-01-19,#1565c0
3-330,category 1,2019-01-19,2019-01-19,#1565c0
1-217.0,category 2,2018-12-27,2018-12-27,#90caf9
3-399.7,moon rising,2019-01-13,2019-01-13,#f44336
8-831.0,sundowner drink,2019-01-17,2019-01-17,#8d6e63
9-984.1,birthday party,2018-12-22,2018-12-22,#90a4ae
F01.9,Meetings,2018-12-26,2018-12-26,#e8a735
```

```
Z71,Meetings,2019-01-12,2019-01-12,#e8a735
B95.7,Meetings,2019-01-15,2019-01-15,#e8a735
T82.7,Meetings,2019-01-15,2019-01-15,#e8a735")

vistime(data)

# ------ It is possible to change all attributes of the timeline using plotly_build(),
# ------ which generates a list which can be inspected using str
p <- vistime(data.frame(event = 1:4, start = c("2019-01-01", "2019-01-10")))
pp <- plotly_build(p) # transform into a list

# Example 1: change x axis font size:
pp$x$layout$xaxis$tickfont <- list(size = 28)
pp

# Example 2: change y axis font size:
pp$x$layout[["yaxis"]]$tickfont <- list(size = 28)
pp

# Example 3: Changing events font size
for (i in 1:length(pp$x$data)) {
  if (pp$x$data[[i]]$mode == "text") pp$x$data[[i]]$textfont$size <- 28
}
pp

# or, using purrr:
text_idx <- which(purrr::map_chr(pp$x$data, "mode") == "text")
for(i in text_idx) pp$x$data[[i]]$textfont$size <- 28
pp

# Example 4: change marker size
# loop over pp$x$data, and change the marker size of all text elements to 50px
for (i in 1:length(pp$x$data)) {
  if (pp$x$data[[i]]$mode == "markers") pp$x$data[[i]]$marker$size <- 40
}
pp

# or, using purrr:
marker_idx <- which(purrr::map_chr(pp$x$data, "mode") == "markers")
for(i in marker_idx) pp$x$data[[i]]$marker$size <- 40
pp

## End(Not run)
```

`vistime_data` *Standardize data to plot on a timeline plot*

Description

Standardize data to plot on a timeline plot

Usage

```
vistime_data(
  data,
  col.event = "event",
  col.start = "start",
  col.end = "end",
  col.group = "group",
  col.color = "color",
  col.fontcolor = "fontcolor",
  col.tooltip = "tooltip",
  optimize_y = TRUE,
  ...
)
```

Arguments

data	data.frame that contains the data to be normalized
col.event	(optional, character) the column name in data that contains event names. Default: *event*.
col.start	(optional, character) the column name in data that contains start dates. Default: *start*.
col.end	(optional, character) the column name in data that contains end dates. Default: *end*.
col.group	(optional, character) the column name in data to be used for grouping. Default: *group*.
col.color	(optional, character) the column name in data that contains colors for events. Default: *color*, if not present, colors are chosen via RColorBrewer.
col.fontcolor	(optional, character) the column name in data that contains the font color for event labels. Default: *fontcolor*, if not present, color will be black.
col.tooltip	(optional, character) the column name in data that contains the mouseover tooltips for the events. Default: *tooltip*, if not present, then tooltips are built from event name and date.
optimize_y	(optional, logical) distribute events on y-axis by smart heuristic (default), otherwise use order of input data.
...	for deprecated arguments up to vistime 1.1.0 (like events, colors, ...)

Value

vistime_data returns a data.frame with the following columns: event, start, end, group, tooltip, label, col, fontcol, subplot, y

Examples

```
# presidents and vice presidents
pres <- data.frame(
  Position = rep(c("President", "Vice"), each = 3),
```

```
  Name = c("Washington", rep(c("Adams", "Jefferson"), 2), "Burr"),
  start = c("1789-03-29", "1797-02-03", "1801-02-03"),
  end = c("1797-02-03", "1801-02-03", "1809-02-03"),
  color = c("#cbb69d", "#603913", "#c69c6e"),
  fontcolor = c("black", "white", "black")
)

vistime_data(pres, col.event = "Position", col.group = "Name")
```

Index