Package ‘vistime’

April 17, 2020

Title Pretty Timelines
Version 1.0.0
Date 2020-04-17
Author Sandro Raabe [aut, cre]
Maintainer Sandro Raabe <sa.ra.online@posteo.de>
Description A library for creating time based charts, like Gantt or timelines. Possible outputs include 'ggplot' diagrams, 'Plotly' graphs and 'data.frame's. Results can be used in the 'RStudio' viewer pane, in 'RMarkdown' documents or in 'Shiny' apps. In the interactive 'Plotly' output, you can hover the mouse pointer over a point or task to show details or drag a rectangle to zoom in.
License GPL-3 | file LICENSE
URL https://shosaco.github.io/vistime/
BugReports https://github.com/shosaco/vistime/issues
Depends R (>= 3.2.0)
Imports assertive (>= 0.1.4), plotly (>= 4.0.0), ggplot2 (>= 2.0.0), RColorBrewer (>= 0.2.2)
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
Suggests knitr, rmarkdown, devtools, testthat, covr, purrr
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2020-04-17 20:20:03 UTC

R topics documented:

  gg_vistime ................................................................. 2
  vistime ................................................................. 3
  vistime_data ............................................................. 6
Index

---

**gg_vistime**

Create a Timeline rendered by ggplot

Description

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

Usage

```r
gg_vistime(
  data,
  events = "event",
  start = "start",
  end = "end",
  groups = "group",
  colors = "color",
  fontcolors = "fontcolor",
  optimize_y = TRUE,
  linewidth = NULL,
  title = NULL,
  show_labels = TRUE,
  background_lines = 10
)
```

Arguments

- **data**
  - data.frame that contains the data to be visualised
- **events**
  - (optional, character) the column name in data that contains event names. Default: `event`.
- **start**
  - (optional, character) the column name in data that contains start dates. Default: `start`.
- **end**
  - (optional, character) the column name in data that contains end dates. Default: `end`.
- **groups**
  - (optional, character) the column name in data to be used for grouping. Default: `group`.
- **colors**
  - (optional, character) the column name in data that contains colors for events. Default: `color`, if not present, colors are chosen via `RColorBrewer`.
- **fontcolors**
  - (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.
**vistime**

Create a Timeline rendered by Plotly

**Description**

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

**Usage**

```r
vistime(
  data,
  events = "event",
  start = "start",
  end = "end",
  groups = "group",
  colors = "color",
  fontcolors = "fontcolor",
  tooltips = "tooltip",
  optimize_y = TRUE,
  linewidth = NULL,
  title = NULL,
  show_labels = TRUE,
  background_lines = 10
)
```

**Examples**

```r
# 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, events = "Position", groups = "Name", title = "Presidents of the USA")
```

**Value**

`gg_vistime` returns an object of class `gg` and `ggplot`.
Arguments

data data.frame that contains the data to be visualised

events (optional, character) the column name in data that contains event names. Default: event.

start (optional, character) the column name in data that contains start dates. Default: start.

dern (optional, character) the column name in data that contains end dates. Default: end.

groups (optional, character) the column name in data to be used for grouping. Default: group.

colors (optional, character) the column name in data that contains colors for events. Default: color, if not present, colors are chosen via RColorBrewer.

fontcolors (optional, character) the column name in data that contains the font color for event labels. Default: fontcolor, if not present, color will be black.

tooltips (optional, character) the column name in data that contains the mouseover tooltips for the events. Default: tooltip, if not present, then tooltips are build 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.

Value

vistime returns an object of class plotly and htmlwidget. See ‘gg_vistime’ for the static ‘ggplot’ version.

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, events = "Position", groups = "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
case 1,2020-12-15,2020-12-24
case 2,2020-12-23,2020-12-29
case 3,2020-12-28,2021-01-06
case 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
case 1,Project,2018-12-22,2018-12-23,#c8e6c9
case 2,Project,2018-12-23,2018-12-29,#a5d6a7
case 3,Project,2018-12-29,2019-01-06,#fb8c00
case 4,Project,2019-01-06,2019-02-02,#dd4b39
Room 334,Team 1,2018-12-22,2018-12-28,#deebe7
Room 335,Team 1,2018-12-28,2019-01-05,#c6dbef
Room 335,Team 1,2019-01-05,2019-01-23,#9ecaee
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,#fb8c00
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)
vistime_data

# Standardize data to plot on a timeline plot

## Description

Standardize data to plot on a timeline plot

## Usage

vistime_data(
  data,
  events = "event",
  start = "start",
  end = "end",
  groups = "group",
  colors = "color",
  fontcolors = "fontcolor",
)
vistime_data

    tooltips = "tooltip",
    optimize_y = TRUE
  )

Arguments

data  data.frame that contains the data to be visualised

events (optional, character) the column name in data that contains event names. Default: event.

start (optional, character) the column name in data that contains start dates. Default: start.

date (optional, character) the column name in data that contains end dates. Default: end.

groups (optional, character) the column name in data to be used for grouping. Default: group.

colors (optional, character) the column name in data that contains colors for events. Default: color, if not present, colors are chosen via RColorBrewer.

fontcolors (optional, character) the column name in data that contains the font color for event labels. Default: fontcolor, if not present, color will be black.

tooltips (optional, character) the column name in data that contains the mouseover tooltips for the events. Default: tooltip, if not present, then tooltips are build from event name and date.

optimize_y (optional, logical) distribute events on y-axis by smart heuristic (default), otherwise use order of input data.

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, events = "Position", groups = "Name")
Index

gg_vistime, 2
vistime, 3
vistime_data, 6