Package ‘echarts4r’

June 17, 2023

Title Create Interactive Graphs with 'Echarts JavaScript' Version 5

Date 2023-06-08

Version 0.4.5

Description
Easily create interactive charts by leveraging the 'Echarts Javascript' library which includes 36 chart types, themes, 'Shiny' proxies and animations.

License Apache License (>= 2.0)

Encoding UTF-8

Imports htmlwidgets, dplyr (>= 0.7.0), purrr, countrycode, broom, shiny, scales, corrplot, htmltools, jsonlite, rstudioapi

Suggests tidyr, testthat, knitr, rmarkdown, covr, data.tree, leaflet, tibble

Depends R (>= 4.1.0)

RoxygenNote 7.2.3

URL https://echarts4r.john-coene.com/
https://github.com/JohnCoene/echarts4r

BugReports https://github.com/JohnCoene/echarts4r/issues/

NeedsCompilation no

Author John Coene [aut, cph],
David Munoz Tord [cre, ctb] (<https://orcid.org/0000-0001-7954-8295>),
Wei Su [ctb],
Helgasoft [ctb],
Xianying Tan [ctb] (<https://orcid.org/0000-0002-6072-3521>),
Robin Cura [ctb] (<https://orcid.org/0000-0001-5926-1828>),
Mathida Chuk [ctb],
Robert Koetsier [ctb] (<https://orcid.org/0000-0002-4477-5401>),
Jelle Geertsma [ctb]

Maintainer David Munoz Tord <david.munoztord@mailbox.org>

Repository CRAN

Date/Publication 2023-06-16 23:30:02 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle_axis</td>
<td>4</td>
</tr>
<tr>
<td>band</td>
<td>5</td>
</tr>
<tr>
<td>band2</td>
<td>6</td>
</tr>
<tr>
<td>callbacks</td>
<td>7</td>
</tr>
<tr>
<td>connections</td>
<td>8</td>
</tr>
<tr>
<td>echarts4r-shiny</td>
<td>10</td>
</tr>
<tr>
<td>echarts4rBox</td>
<td>11</td>
</tr>
<tr>
<td>echarts4rBoxOutput</td>
<td>13</td>
</tr>
<tr>
<td>e_animation</td>
<td>13</td>
</tr>
<tr>
<td>e_append1_p</td>
<td>14</td>
</tr>
<tr>
<td>e_area</td>
<td>16</td>
</tr>
<tr>
<td>e_aria</td>
<td>18</td>
</tr>
<tr>
<td>e_axis</td>
<td>18</td>
</tr>
<tr>
<td>e_axis_3d</td>
<td>20</td>
</tr>
<tr>
<td>e_axis_labels</td>
<td>22</td>
</tr>
<tr>
<td>e_axis_pointer</td>
<td>22</td>
</tr>
<tr>
<td>e_axis_stagger</td>
<td>23</td>
</tr>
<tr>
<td>e_bar</td>
<td>23</td>
</tr>
<tr>
<td>e_bar_3d</td>
<td>25</td>
</tr>
<tr>
<td>e_boxplot</td>
<td>27</td>
</tr>
<tr>
<td>e_brush</td>
<td>28</td>
</tr>
<tr>
<td>e_button</td>
<td>29</td>
</tr>
<tr>
<td>e_calendar</td>
<td>30</td>
</tr>
<tr>
<td>e_candle</td>
<td>31</td>
</tr>
<tr>
<td>e_capture</td>
<td>32</td>
</tr>
<tr>
<td>e_cloud</td>
<td>33</td>
</tr>
<tr>
<td>e_color</td>
<td>34</td>
</tr>
<tr>
<td>e_color_range</td>
<td>35</td>
</tr>
<tr>
<td>e_common</td>
<td>36</td>
</tr>
<tr>
<td>e_correlations</td>
<td>36</td>
</tr>
<tr>
<td>e_country_names</td>
<td>37</td>
</tr>
<tr>
<td>e_datazoom</td>
<td>37</td>
</tr>
<tr>
<td>e_dims</td>
<td>38</td>
</tr>
<tr>
<td>e_dispatch_action_p</td>
<td>39</td>
</tr>
<tr>
<td>e_draft</td>
<td>40</td>
</tr>
<tr>
<td>e_draw_p</td>
<td>40</td>
</tr>
<tr>
<td>e_error_bar</td>
<td>41</td>
</tr>
<tr>
<td>e_execute</td>
<td>43</td>
</tr>
<tr>
<td>e_facet</td>
<td>43</td>
</tr>
<tr>
<td>e_flip_coords</td>
<td>44</td>
</tr>
<tr>
<td>e_flow_gl</td>
<td>45</td>
</tr>
<tr>
<td>e_focus_adjacency_p</td>
<td>48</td>
</tr>
<tr>
<td>e_format_axis_p</td>
<td>50</td>
</tr>
<tr>
<td>e_funnel</td>
<td>51</td>
</tr>
<tr>
<td>e_gauge</td>
<td>52</td>
</tr>
<tr>
<td>e_geo</td>
<td>53</td>
</tr>
</tbody>
</table>
angle_axis

Description
Customise angle axis.

Usage

\[
e_{\text{angle\_axis}}(e, \text{serie}, \text{show} = \text{TRUE}, \ldots)
\]

\[
e_{\text{angle\_axis\_}}(e, \text{serie} = \text{NULL}, \text{show} = \text{TRUE}, \ldots)
\]

Arguments

- \text{e} \quad \text{An echart\_4r object as returned by e\_charts or a proxy as returned by echart\_4rProxy.}
- \text{serie} \quad \text{Serie to use as axis labels.}
- \text{show} \quad \text{Whether to display the axis.}
- \ldots \quad \text{Any other option to pass, check See Also section.}
See Also

Additional arguments

Examples

```r
df <- data.frame(x = 1:100, y = seq(1, 200, by = 2))

df |> 
  e_charts(x) |> 
  e_polar(FALSE) |> 
  e_angle_axis(FALSE) |> 
  e_radius_axis(FALSE) |> 
  e_line(y, coord.system = "polar", smooth = TRUE) |> 
  e_legend(show = FALSE)

df <- data.frame(x = LETTERS[1:5], y = runif(5))

df |> 
  e_charts(x) |> 
  e_polar() |> 
  e_angle_axis(x) |> 
  e_radius_axis() |> 
  e_line(y, coord.system = "polar", smooth = TRUE)
```

---

**band**  

Confidence bands

Description

Add confidence bands

Usage

```r
e_band(
  e,
  min,
  max,
  stack = "confidence-band",
  symbol = c("none", "none"),
  areaStyle = list(list(color = "rgba(0,0,0,0)"), list()),
  legend = list(FALSE, FALSE),
  ...
)

e_band_(
  e,
  min,
  max,
```
stack = "confidence-band",
symbol = c("none", "none"),
areaStyle = list(list(color = "rgba(0,0,0,0)"), list()),
legend = list(FALSE, FALSE),
...
)

Arguments

e An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.

min, max series.

stack Name of stack.

symbol Whether to show symbols on lower and upper band lines.

areaStyle The style of lower and upper bands, i.e.: color.

legend Whether to show min and max in legend.

... All options must be of vectors or lists of length 2 where the first argument is for the lower bound and the second for the upper bound, see examples.

Examples

df <- data.frame(
  x = 1:10,
  y = runif(10, 5, 10)
) |> 
dplyr::mutate(
  lwr = y - runif(10, 1, 3),
  upr = y + runif(10, 2, 4)
) 

df |> 
e_charts(x) |> 
e_line(y) |> 
e_band(lwr, upr)

---

**Area bands**

**Description**

Add area bands

**Usage**

```r
e_band2(e, lower, upper, ...)
```

```r
e_band2_(
  e,
```
lower,
upper,
name = NULL,
legend = TRUE,
y_index = 0,
x_index = 0,
coord_system = "cartesian2d",
itemStyle = list.borderWidth = 0.5),
...
)

Arguments

e An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
lower, upper series of lower and upper borders of the band
... additional options
name name of the serie.
legend Whether to add serie to legend.
x_index, y_index Indexes of x and y axis.
coord_system Coordinate system to plot against.
itemStyle mostly used for borderWidth, default 0.5

Examples

data(EuStockMarkets)
as.data.frame(EuStockMarkets) |> 
dplyr::slice_head(n = 200) |> 
dplyr::mutate(day = 1:dplyr::n()) |> 
e_charts(day) |> 
e_line(CAC, symbol = "none") |> 
e_band2(DAX, FTSE, color = "lemonchiffon") |> 
e_band2(DAX, SMI, color = "lightblue", itemStyle = list(borderWidth = 0)) |> 
e_y_axis(scale = TRUE) |> 
e_datazoom(start = 50)

callbacks

Callbacks

Description

Binds events to chart interactions.

Usage

e_on(e, query, handler, event = "click")
e_off(e, query, handler, event = "click")
Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **query**: Condition that triggers the handler.
- **handler**: JavaScript handler, passed to `JS`.
- **event**: Event that triggers the handler.

See Also

- [official documentation](#)

Examples

```r
cars |> e_charts(speed) |> e_scatter(dist) |> e_on(
  list(seriesName = "dist"),
  "function(){alert('Serie clicked')}"
)
```

---

connections | Connect charts

Description

Connect charts together.

Usage

- `e_connect(e, ids)`
- `e_group(e, group)`
- `e_connect_group(e, group)`
- `e_disconnect_group(e, group = NULL)`
- `e_arrange(..., rows = NULL, cols = NULL, width = "xs", title = NULL)`

Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **ids**: Scalar, vector or list of ids of chart to connect with.
- **group**: Group name.
- **...**: Any `echarts` objects.
- **rows, cols**: Number of rows and columns.
- **width**: Width of columns, one of `xs`, `md`, `lg`.
- **title**: Title of charts.
Value

e_arrange: in an interactive session, returns a htmltools::browsable, in rmarkdown returns a container (htmltools::div).

Functions

• e_connect: connects charts by ids, cannot be disconnected.
• e_group: assigns a group to chart.
• e_connect_group: connects chart with another group.
• e_disconnect_group: disconnects chart from group.
• e_arrange: arrange charts.

Note

e_arrange may not work properly in the RStudio viewer.

Examples

# linked datazoom
e1 <- cars |>  
e_charts(  
speed,  
height = 200  
) |>  
e_scatter(dist) |>  
e_datazoom(show = FALSE) |>  
e_group("grp") # assign group

e2 <- cars |>  
e_charts(  
dist,  
height = 200  
) |>  
e_scatter(speed) |>  
e_datazoom() |>  
e_group("grp") # assign group  
e_connect_group("grp") # connect

if (interactive()) {
  e_arrange(e1, e2, title = "Linked datazoom")
}
echarts4r-shiny  Shiny bindings for echarts4r

Description
Output and render functions for using echarts4r within Shiny applications and interactive Rmd documents.

Usage

```r
echarts4rOutput(outputId, width = "100\%", height = "400px")
renderEcharts4r(expr, env = parent.frame(), quoted = FALSE)

echarts4rProxy(
  id,
  data,
  x,
  timeline = FALSE,
  session = shiny::getDefaultReactiveDomain(),
  reorder = TRUE
)

ecarts4r_proxy(
  id,
  data,
  x,
  timeline = FALSE,
  session = shiny::getDefaultReactiveDomain(),
  reorder = TRUE
)
```

Arguments

- `outputId`: output variable to read from.
- `width`, `height`: Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
- `expr`: An expression that generates a echarts4r
- `env`: The environment in which to evaluate expr.
- `quoted`: Is expr a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.
- `id`: Target chart id.
- `data`: A data.frame.
- `x`: Column name containing x axis.
- `timeline`: Set to TRUE to build a timeline, see timeline section.
session Shiny session.
reorder Set the FALSE to not reorder numeric x axis values.

Callbacks

• id_brush: returns data on brushed data points.
• id_legend_change: returns series name of legend selected/unselected.
• id_clicked_data: returns data of clicked data point.
• id_clicked_data_value: returns value of clicked data point.
• id_clicked_row: returns row number of clicked data point.
• id_clicked_serie: returns name of serie of clicked data point.
• id_mouseover_data: returns data on hovered data point.
• id_mouseover_data_value: returns value of hovered data point.
• id_mouseover_row: returns row of hovered data point.
• id_mouseover_serie: returns name of serie of hovered data point.

Proxies

The echarts4rProxy function returns a proxy for chart which allows manipulating a drawn chart, adding data, adding or removing series, etc. without redrawing the entire chart.

• e_append1_p & e_append2_p
• e_showtip_p & e_hidetip_p
• e_highlight_p & e_downplay_p
• e_focus_adjacency & e_unfocus_adjacency
• e_dispatch_action_p
• e_execute
• e_remove_serie_p

Description

Renders a data box in shiny.
Usage

echarts4rBox(
  data,
  x,
  y,
  text = "",
  subtext = "",
  type = c("bar", "line", "scatter", "area", "step"),
  ..., 
  color = "#ffffff",
  text_color = "#ffffff",
  background_color = "#293c55",
  step = c("start", "middle", "end"),
  title_args = list(),
  tooltip = list(trigger = "axis")
)

Arguments

data: A dataframe containing data to plot.

x, y: Bare column name of variables to draw.

text, subtext: Title and subtitle of box.

type: Chart type to draw.

...: Additional arguments to pass to the serie.

color: Color of chart in box.

text_color: Color of text.

background_color: Color of box.

step: Step method, only used if type = "step".

title_args: Additional arguments to add to the title.

tooltip: Tooltip to use.

See Also

renderEcharts4rBox, echarts4rBoxOutput

Examples

library(shiny)

ui <- fluidPage(
  fluidRow(
    column(3, echarts4rBoxOutput("box1"))
  )
)
server <- function(input, output) {
  output$box1 <- renderEcharts4rBox({
    echarts4rBox(cars, speed, dist, "Cars", type = "bar")
  })
}
## Not run:
shinyApp(ui, server)
## End(Not run)

**echarts4rBoxOutput**  
*Box Output*

**Description**
Place box output in Shiny ui.

**Usage**
```r
echarts4rBoxOutput(id, height = 150)
```

**Arguments**
- `id`: Id of box.
- `height`: Height of box, any valid CSS value, numerics are treated as pixels.

**e_animation**  
*Animation*

**Description**
Customise animations.

**Usage**
```r
e_animation(
  e,
  show = TRUE,
  threshold = NULL,
  duration = NULL,
  easing = NULL,
  delay = NULL,
  duration.update = NULL,
  easing.update = NULL,
  delay.update = NULL
)
```
### Arguments

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **show**: Set to show animation.
- **threshold**: Whether to set graphic number threshold to animation. Animation will be disabled when graphic number is larger than threshold.
- **duration**: Duration of the first animation.
- **easing**: Easing method used for the first animation.
- **delay**: Delay before updating the first animation.
- **duration.update**: Time for animation to complete.
- **easing.update**: Easing method used for animation.
- **delay.update**: Delay before updating animation.

### See Also

- Additional arguments

### Examples

```r
mtcars |>  
  e_charts(mpg) |>  
  e_area(drat) |>  
  e_animation(duration = 10000)
```

---

### e_append1_p

**Append Proxy**

**Append data dynamically.**

**Usage**

```r
e_append1_p(proxy, series_index = NULL, data, x, y, name = NULL)
e_append1_p_(proxy, series_index = NULL, data, x, y, name = NULL)
e_append2_p(  
  proxy,  
  series_index = NULL,  
  data,  
  x,  
  y,  
  z,  
  scale = NULL,
```

**Description**

Append data dynamically.
symbol_size = 1
)

e_append2_p_(
  proxy,
  series_index = NULL,
  data,
  x,
  y,
  z,
  scale = NULL,
  symbol_size = 1
)

Arguments

proxy     An echarts4r proxy as returned by echarts4rProxy.
series_index Index of serie to append to (starts from 0).
data     Data.frame containing data to append.
x, y, z     Columns names to plot.
nname if using ‘bind’ with e.g ‘e_scatter’ this can be used to supply the colname for the
          name attribute bind is mapping to
scale     A scaling function as passed to e_scatter.
symbol_size Multiplier of scaling function as in e_scatter.

Details

Currently not all types of series supported incremental rendering when using appendData. Only
these types of series support it: e_scatter and e_line of pure echarts, and e_scatter_3d, and
e_line_3d of echarts-gl.

Examples

## Not run:
library(shiny)

ui <- fluidPage(
  actionButton("add", "Add Data to y"),
  echarts4rOutput("plot"),
  h4("Brush"),
  verbatimTextOutput("selected"),
  h4("Legend select change"),
  verbatimTextOutput("legend")
)

server <- function(input, output, session) {
  data <- data.frame(x = rnorm(10, 5, 3), y = rnorm(10, 50, 12), z = rnorm(10, 5, 20))

  react <- eventReactive(input$add, {

```
set.seed(sample(1:1000, 1))
data.frame(x = rnorm(10, 5, 2), y = rnorm(10, 50, 10), z = rnorm(10, 5, 20))
```

```r
output$plot <- renderEcharts4r({
data |>
e_charts(x) |>
e_scatter(y, z, scale = NULL) |>
e_scatter(z) |>
e_brush()
})
```

```r
observeEvent(input$add, {
echarts4rProxy("plot") |>
e_append2_p(0, react(), x, y, z)
})
```

```r
output$selected <- renderPrint({
input$plot_brush
})
```

```r
output$legend <- renderPrint({
input$plot_legend_change
})
```

shinyApp(ui, server)
```

## e_area

### Description
Add area serie. Note that this is NOT an unique series type. Rather, this function is a shorthand for using `e_bar()` with `areaStyle = list()` enabled.

### Usage
```
e_area(
e,
serie,
bind,
name = NULL,
legend = TRUE,
y_index = 0,
x_index = 0,
)```
coord_system = "cartesian2d",
...
)

e_area_(
  e,
  serie,
  bind = NULL,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)

Arguments

e An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
serie Column name of serie to plot.
bind Binding between datasets, namely for use of `e_brush`.
name name of the serie.
legend Whether to add serie to legend.
x_index, y_index Indexes of x and y axis.
coord_system Coordinate system to plot against.
...
Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

```r
CO2 |>
  group_by(Plant) |>
  e_charts(conc) |>
  e_area(uptake) |>
  e_tooltip(trigger = "axis")

# timeline
iris |>
  group_by(Species) |>
  e_charts(Sepal.Length, timeline = TRUE) |>
  e_area(Sepal.Width) |>
  e_tooltip(trigger = "axis")
```
**e_aria**

### Description

W3C defined the Accessible Rich Internet Applications Suite (WAI-ARIA) to make Web content and Web applications more accessible to the disabled. From ECharts 4.0, echarts4r supports ARIA by generating description for charts automatically.

### Usage

e_aria(e, enabled = TRUE, ...)

### Arguments

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **enabled**: Whether to enable aria helper text.
- **...**: Any other option to pass, check See Also section.

### Details

There should be an aria-label attribute on the chart DOM, which can help the disabled understand the content of charts with the help of certain devices.

### See Also

[official documentation](#)

---

**e_axis**

### Description

Customise axis.

### Usage

```r

e_axis(
e, serie, 
axis = c("x", "y", "z"), 
index = 0, 
formatter = NULL, 
margin = 0, 
...
```

---
e_axis

)

e_axis_(
  e,
  serie = NULL,
  axis = c("x", "y", "z"),
  index = 0,
  formatter = NULL,
  margin = 0,
  ...
)

e_x_axis_(e, serie = NULL, index = 0, formatter = NULL, margin = 0, ...)
e_y_axis_(e, serie = NULL, index = 0, formatter = NULL, margin = 0, ...)
e_z_axis_(e, serie = NULL, index = 0, margin = 0, ...)
e_x_axis(e, serie, index = 0, formatter = NULL, margin = 0, ...)
e_y_axis(e, serie, index = 0, formatter = NULL, margin = 0, ...)
e_z_axis(e, serie, index = 0, margin = 0, ...)
e_rm_axis(e, axis = c("x", "y", "z"))
e_axis_formatter(
  style = c("decimal", "percent", "currency"),
  digits = 0,
  locale = NULL,
  currency = "USD"
)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie Column name of serie to range the axis. If used the range of the serie is used as, min an max.
axis Axis to customise.
index Index of axis to customise.
formatter An axis formatter as returned by e_axis_formatter.
margin Margin to apply to serie: min = serie−margin and max = serie+margin
... Any other option to pass, check See Also section.
style Formatter style, one of decimal, percent, or currency.
digits Number of decimals.
locale Locale, if NULL then it is inferred from Sys.getlocale.
currency Currency to to display.
**e_axis_3d**

**Axis 3D**

**Description**

Customise 3D axis.

**Details**

The `e_axis_formatter` may not work in RStudio, open the plot in your browser. It will display just fine in Rmarkdown and Shiny.

**Functions**

- `e_axis` to customise axis
- `e_rm_axis` to remove axis

**See Also**

Additional x arguments, Additional y arguments

**Examples**

```r
# range axis based on serie
cars |>  
  e_charts(speed) |>  
  e_line(dist) |>  
  e_x_axis(speed) |>  
  e_y_axis(dist)

# use formatter
cars |>  
  dplyr::mutate(  
    speed = speed / 25  
  ) |>  
  e_charts(speed) |>  
  e_scatter(dist) |>  
  e_y_axis(  
    formatter = e_axis_formatter("currency")  
  ) |>  
  e_x_axis(  
    formatter = e_axis_formatter("percent", digits = 0)  
  )

# plot all labels & rotate
USArrests |>  
  head(10) |>  
  tibble::rownames_to_column(var = "State") |>  
  e_charts(State) |>  
  e_area(Murder) |>  
  e_x_axis(axisLabel = list(interval = 0, rotate = 45)) # rotate
```

---

**e_axis_3d**
Usage

```r
e_axis_3d(e, axis = c("x", "y", "z"), index = 0, ...)
e_x_axis_3d(e, index = 0, ...)
e_y_axis_3d(e, index = 0, ...)
e_z_axis_3d(e, index = 0, ...)
```

Arguments

- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `axis` Axis to customise.
- `index` Index of axis to customise.
- `...` Any other option to pass, check See Also section.

See Also

`Additional x arguments, Additional y arguments, Additional z arguments`

Examples

```r
# phony data
v <- LETTERS[1:10]
matrix <- data.frame(
  x = sample(v, 300, replace = TRUE),
  y = sample(v, 300, replace = TRUE),
  z1 = rnorm(300, 10, 1),
  z2 = rnorm(300, 10, 1),
  stringsAsFactors = FALSE
) |> 
  dplyr::group_by(x, y) |> 
  dplyr::summarise(
    z1 = sum(z1),
    z2 = sum(z2)
  ) |> 
  dplyr::ungroup()

trans <- list(opacity = 0.4) # transparency
emphasis <- list(itemStyle = list(color = "#313695"))

matrix |> 
  e_charts(x) |> 
  e_bar_3d(y, z1, stack = "stack", name = "Serie 1", itemStyle = trans, emphasis = emphasis) |> 
  e_bar_3d(y, z2, stack = "stack", name = "Serie 2", itemStyle = trans, emphasis = emphasis) |> 
  e_x_axis_3d(axisLine = list(lineStyle = list(color = "blue")))
```
**e_axis_labels**  
*Axis Labels*

Description

Convenience function to add axis labels.

Usage

```r
e_axis_labels(e, x = "", y = "")
```

Arguments

- `e` An `echarts4r` object as returned by `eCharts` or a proxy as returned by `echarts4rProxy`.
- `x, y` Labels of axes.

Examples

```r
cars |>
  e_charts(speed) |>
  e_scatter(dist) |>
  e_axis_labels(
    x = "speed",
    y = "distance"
  )
```

**e_axis_pointer**  
*Axis Pointer*

Description

Customise axis pointer.

Usage

```r
e_axis_pointer(e, ...)
```

Arguments

- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `...` Any other option to pass, check See Also section.

See Also

*Additional arguments*
### e_axis_stagger

**Stagger Axis Labels**

**Description**

Stagger axis labels.

**Usage**

```r
e_axis_stagger(e)
```

**Arguments**

- `e`
  
  An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.

**Examples**

```r
df <- data.frame(
  x = c("a very long label", "Another long label"),
  y = 1:2
)

df |> 
  e_charts(x, width = 150) |> 
  e_bar(y) |> 
  e_axis_stagger()
```

---

### e_bar

**Bar and Line chart**

**Description**

Add bar serie.

**Usage**

```r
e_bar(
  e,
  serie,
  bind,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)```
e_bar(
  e,
  serie,
  bind = NULL,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)

Arguments

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **serie**: Column name of serie to plot.
- **bind**: Binding between datasets, namely for use of `e_brush`.
- **name**: name of the serie.
- **legend**: Whether to add serie to legend.
- **x_index, y_index**: Indexes of x and y axis.
- **coord_system**: Coordinate system to plot against.

... Any other option to pass, check See Also section.

Note

The bar serie expects the data on the x axis to be categorical in R this means a factor or character. If the data on the x axis is numeric everything should work well in most cases but strange behaviour may be observed.

See Also

Additional arguments

Examples

```r
library(dplyr)
mtcars |>
  tibble::rownames_to_column("model") |>
  mutate(total = mpg + qsec) |>
  arrange(desc(total)) |>
  e_charts(model) |>
  e_bar(mpg, stack = "grp") |>
  e_bar(qsec, stack = "grp")
```
Description

Add 3D bars

Usage

e_bar_3d( 
  e, 
  y, 
  z, 
  bind, 
  coord_system = "cartesian3D", 
  name = NULL, 
  rm_x = TRUE, 
  rm_y = TRUE, 
  ... 
)

e_bar_3d_( 
  e, 
  y, 
  z, 
  bind = NULL, 
  coord_system = "cartesian3D", 
  name = NULL, 
  rm_x = TRUE, 
  rm_y = TRUE, 
  ... 
)

Arguments

e Bar 3D

An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
y, z Coordinates.
bind Binding.
coord_system Coordinate system to use, one of cartesian3D, geo3D, globe.
name name of the serie.
rm_x, rm_y Whether to remove x and y axis, defaults to TRUE.
... Any other option to pass, check See Also section.

See Also

Additional arguments
Examples

## Not run:
# volcano
volcano |> as.table() |> as.data.frame() |> dplyr::mutate(`Var1` = as.integer(`Var1`), `Var2` = as.integer(`Var2`)) |> e_charts(`Var1`) |> e_bar_3d(`Var2`, Freq) |> e_visual_map(Freq)

url <- paste0("https://echarts.apache.org/examples/", "data-gl/asset/data/population.json")
data <- jsonlite::fromJSON(url)
data <- as.data.frame(data)
names(data) <- c("lon", "lat", "value")

# globe
data |
| e_charts(`lon`) |
| e_globe() |
| e_bar_3d(`lat`, `value`, coord_system = "globe") |
| e_visual_map()

# get3d
data |
| e_charts(`lon`) |
| e_geo_3d() |
| e_bar_3d(`lat`, `value`, coord_system = "geo3D") |
| e_visual_map()

# stacked
v <- LETTERS[1:10]
matrix <- data.frame(`x` = sample(v, 300, replace = TRUE), `y` = sample(v, 300, replace = TRUE), `z1` = rnorm(300, 10, 1), `z2` = rnorm(300, 10, 1), stringsAsFactors = FALSE) |
| dplyr::group_by(`x`, `y`) |
| dplyr::summarise(`z1` = sum(`z1`), `z2` = sum(`z2`)) |
| dplyr::ungroup()
trans <- list(opacity = 0.4) # transparency
emphasis <- list(itemStyle = list(color = "#313695"))

matrix |>  
  e_charts(x) |>  
  e_bar_3d(y, z1, stack = "stack", name = "Serie 1", itemStyle = trans, emphasis = emphasis) |>  
  e_bar_3d(y, z2, stack = "stack", name = "Serie 2", itemStyle = trans, emphasis = emphasis) |>  
  e_legend()

# timeline
matrix |>  
  group_by(x) |>  
  e_charts(y, timeline = TRUE) |>  
  e_bar_3d(z1, z2) |>  
  e_visual_map(z2)

## End(Not run)

---

### e_boxplot

**Boxplot**

**Description**

Draw boxplot.

**Usage**

```r
e_boxplot(e, serie, name = NULL, outliers = TRUE, ...)
e_boxplot_(e, serie, name = NULL, outliers = TRUE, ...)
```

**Arguments**

- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `serie` Column name of serie to plot.
- `name` name of the serie.
- `outliers` Whether to plot outliers.
- `...` Any other option to pass, check See Also section.

**See Also**

Additional arguments
Examples

```
df <- data.frame(
  x = c(1:10, 25),
  y = c(1:10, -6)
)

df |>  
e_charts() |>  
e_boxplot(y, outliers = TRUE) |>  
e_boxplot(x, outliers = TRUE)
```

---

### e_brush

#### Brush

**Description**

Add a brush.

**Usage**

```
e_brush(e, x_index = NULL, y_index = NULL, brush_link = "all", ...)
```

**Arguments**

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **x_index, y_index**: Indexes of x and y axis.
- **brush_link**: Links interaction between selected items in different series.
- **...**: Any other option to pass, check See Also section.

**brush_link**

$\$

- c(3, 4, 5), for interacting series with seriesIndex as 3, 4, or 5.
- all, for interacting all series.
- none, for disabling.

**See Also**

Additional arguments
**e_button**

**Examples**

```r
quakes |>
  e_charts(long) |>
  e_geo(
    boundingCoords = list(
      c(190, -10),
      c(180, -40)
    )
  ) |>
  e_scatter(lat, mag, stations, coord.system = "geo", name = "mag") |>
  e_data(quakes, depth) |>
  e_scatter(mag, mag, stations, name = "mag & depth") |>
  e_grid(right = 40, top = 100, width = "30%") |>
  e_y_axis(type = "value", name = "depth", min = 3.5) |>
  e_brush() |>
  e_theme("dark")
```

---

**Description**

Add a button to your visualisation.

**Usage**

```r
e_button(e, id, ..., position = "top", tag = htmltools::tags$button)
```

**Arguments**

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **id**: A valid CSS id.
- **...**: Content of the button, compliant with `htmltools`.
- **position**: Position of button, top or bottom.
- **tag**: A valid `htmltools::tags` function.

**Examples**

```r
iris |>
  group_by(Species) |>
  e_charts(Sepal.Length) |>
  e_line(Sepal.Width) |>
  e_line(Petal.Length) |>
  e_highlight(series_name = "setosa", btn = "myBtn") |>
  e_button("myBtn", "highlight stuff")
```
e_calendar

Description
Calendar

Usage
\[
e\_calendar(e, range, ...)\]

Arguments
- `e`: An `echarts4r` object as returned by `e\_charts` or a proxy as returned by `echarts4rProxy`.
- `range`: Range of calendar format, string or vector.
- `...`: Any other option to pass, check See Also section.

See Also
Additional arguments

Examples
```r
dates <- seq.Date(as.Date("2017-01-01"), as.Date("2019-12-31"), by = "day")
values <- rnorm(length(dates), 20, 6)
year <- data.frame(date = dates, values = values)

year |> e\_charts(date) |> e\_calendar(range = "2017") |> e\_visual_map(max = 30) |> e\_heatmap(values, coord_system = "calendar")

# month
year |> e\_charts(date) |> e\_calendar(range = "2017-01") |> e\_visual_map(max = 30) |> e\_heatmap(values, coord_system = "calendar")

# range
year |> e\_charts(date) |> e\_calendar(range = c("2018-01", "2018-07")) |> e\_visual_map(max = 30) |> e\_heatmap(values, coord_system = "calendar")
```
Description
Add a candlestick chart.

Usage
```r
e_candle(e, opening, closing, low, high, bind, name = NULL, legend = TRUE, ...)
```

```r
e_candle_(
  e,
  opening,
  closing,
  low,
  high,
  bind = NULL,
  name = NULL,
  legend = TRUE,
  ...
)
```

Arguments
- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `opening`, `closing`, `low`, `high`: Stock prices.
- `bind`: Binding between datasets, namely for use of `e_brush`.
- `name`: Name of the serie.
- `legend`: Whether to add serie to legend.
- `...`: Any other option to pass, check See Also section.

See Also
- `Additional arguments`

Examples
```r
date <- c(
  "2017-01-01",
  "2017-01-02",
  "2017-01-03",
  "2017-01-04",
  "2017-03-05",
  "2017-01-06",
  "2017-01-07"
)
```r
stock <- data.frame(
  date = date,
  opening = c(200.60, 200.22, 198.43, 199.05, 203.54, 203.40, 208.34),
  closing = c(200.72, 198.85, 199.05, 203.73, 204.08, 208.11, 211.88),
  low = c(197.82, 198.07, 197.90, 198.10, 202.00, 201.50, 207.60),
  high = c(203.32, 200.67, 200.00, 203.95, 204.90, 208.44, 213.17)
)
```

```r
e_charts(date) |>
e_candle(opening, closing, low, high) |>
e_y_axis(min = 190, max = 220)
```

---

### e_capture

<table>
<thead>
<tr>
<th>Capture event</th>
</tr>
</thead>
</table>

#### Description

Add an event capture.

#### Usage

```r
e_capture(e, event)
```

#### Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **event**: An event name from the event documentation.

#### Details

Many events can be captured, however not all are integrated, you can pass one that is not implemented with this function.

#### Examples

```r
## Not run:
# add datazoom
library(shiny)

ui <- fluidPage(
  echarts4rOutput("chart"),
 verbatimTextOutput("zoom")
)

server <- function(input, output) {
  output$chart <- renderEcharts4r({
```
mtcars |> 
  e_charts(mpg) |> 
  e_scatter(qsec) |> 
  e_datazoom() |> 
  e_capture("datazoom")
})

output$zoom <- renderPrint({
  input$chart_datazoom
})

if (interactive()) {
  shinyApp(ui, server)
}

## End(Not run)

---

e_cloud | **Wordcloud**

---

**Description**

Draw a wordcloud.

**Usage**

e_cloud(e, word, freq, color, rm_x = TRUE, rm_y = TRUE, ...)
e_cloud_(e, word, freq, color = NULL, rm_x = TRUE, rm_y = TRUE, ...)

**Arguments**

e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
word, freq | Terms and their frequencies.
color | Word color.
rm_x, rm_y | Whether to remove x and y axis, defaults to TRUE.
... | Any other option to pass, check See Also section.

**See Also**

official documentation
Examples

```r
words <- function(n = 5000) {
  a <- do.call(paste0, replicate(5, sample(LETTERS, n, TRUE), FALSE))
  paste0(a, sprintf("%04d", sample(9999, n, TRUE)), sample(LETTERS, n, TRUE))
}

tf <- data.frame(
  terms = words(100),
  freq = rnorm(100, 55, 10)
) |> 
  dplyr::arrange(-freq)

tf |> 
  e_color_range(freq, color) |>
  e_charts() |>
  e_cloud(terms, freq, color, shape = "circle", sizeRange = c(3, 15))
```

---

e_color | Color
---

Description

Customise chart and background colors.

Usage

```r
e_color(e, color = NULL, background = NULL, append = TRUE)
```

## S3 method for class 'echarts4r'
```r
e_color(e, color = NULL, background = NULL, append = TRUE)
```

## S3 method for class 'echarts4rProxy'
```r
e_color(e, color = NULL, background = NULL, append = TRUE)
```

Arguments

- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `color` Vector of colors.
- `background` Background color.
- `append` Only applicable to `echarts4rProxy`. Whether to append the ‘color’ to the existing array (vector) or colors or to replace it.

See Also

`e_theme`, Official color documentation, Official background documentation
### e_color_range

**Examples**

```
mtcars |>  
  e_charts(drat) |>  
  e_line(mpg) |>  
  e_area(qsec) |>  
  e_color(  
    c("red", "blue"),  
    "#d3d3d3"  
  )  
```

<table>
<thead>
<tr>
<th>e_color_range</th>
<th>Color range</th>
</tr>
</thead>
</table>

**Description**

Build manual color range

**Usage**

```
e_color_range(  
  data,  
  input,  
  output,  
  colors = c("#bf444c", "#d88273", "#f6efa6"),  
  ...  
)  
```

```
e_color_range_(  
  data,  
  input,  
  output,  
  colors = c("#bf444c", "#d88273", "#f6efa6"),  
  ...  
)  
```

**Arguments**

- `data` Data.frame in which to find column names.
- `input`, `output` Input and output columns.
- `colors` Colors to pass to `colorRampPalette`.
- `...` Any other argument to pass to `colorRampPalette`.

**Examples**

```
df <- data.frame(val = 1:10)  
```

```
e_color_range(df, val, colors)  
```
### e_common

**General options**

**Description**
General options

**Usage**

```r
e_common(font_family = NULL, theme = NULL)
```

**Arguments**

- `font_family`: Font family.
- `theme`: A theme.

### e_correlations

**Correlation**

**Description**
Correlation

**Usage**

```r
e_correlations(e, order = NULL, visual_map = TRUE, ...)
```

**Arguments**

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `order`: Ordering method, passed to `corrMatOrder`.
- `visual_map`: Whether to add the visual map.
- `...`: Any argument to pass to `e_heatmap` and `e_visual_map`.

**Examples**

```r
cor(mtcars) |>
e_charts() |>
e_correlations(
  order = "hclust",
  visual_map = FALSE
) |>
e_visual_map(
  min = -1,
  max = 1
)
```
e_country_names  

Country names

Description
Convert country names to echarts format.

Usage

```r
e_country_names(data, input, output, type = "iso2c", ...)
e_country_names_(data, input, output = NULL, type = "iso2c", ...)
```

Arguments
- `data` Data.frame in which to find column names.
- `input, output` Input and output columns.
- `type` Passed to `countrycode` `origin` parameter.
- `...` Any other parameter to pass to `countrycode`.

Details
Taiwan and Hong Kong cannot be plotted.

Examples
```r
cns <- data.frame(country = c("US", "BE"))
# replace
e_country_names(cns, country)
# specify output
e_country_names(cns, country, country_name)
```

e_datazoom  

Data zoom

Description
Add data zoom.

Usage

```r
e_datazoom(e, x_index = NULL, y_index = NULL, toolbox = TRUE, ...)
```
Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **x_index, y_index**: Indexes of x and y axis.
- **toolbox**: Whether to add the toolbox, `e_toolbox_feature`, (e_toolbox_feature(e, “dataZoom”)).
- **...**: Any other option to pass, check See Also section.

See Also

- Additional arguments

Examples

```r
USArrests |>
  e_charts(UrbanPop) |>
  e_line(Assault) |>
  e_area(Murder, y_index = 1, x_index = 1) |>
  e_y_axis(gridIndex = 1) |>
  e_x_axis(gridIndex = 1) |>
  e_grid(height = "35%") |>
  e_grid(height = "35%", top = "50%") |>
  e_toolbox_feature("dataZoom", title = list(zoom = "zoom", back = "back")) |>
  e_datazoom(x_index = c(0, 1))
```

---

### e_dims

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
</table>

Description

Sets the dimensions of the chart _internally_. This will only affect the dimensions of the chart within its parent container. Use the 'height' and 'width' arguments of `[e_charts]` if you want to change the dimensions of said parent (recommended).

Usage

```r
e_dims(e, height = "auto", width = "auto")
```

Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **height, width**: Dimensions in pixels, percentage or string.
**Description**

Create your own proxies, essentially a wrapper around the action API.

**Usage**

\[
e\_\text{dispatch}\_\text{action}_p(\text{proxy}, \text{type}, \ldots)
\]

**Arguments**

- `proxy` An echarts4r proxy as returned by `echarts4rProxy`.
- `type` Type of action to dispatch, i.e.: `highlight`.
- `...` Named options.

**Examples**

```r
## Not run:
library(shiny)

ui <- fluidPage(
  fluidRow(
    column(8, echarts4rOutput("chart")),
    column(4, actionButton("zoom", "Zoom"))
  )
)

cars |>
e\_\text{charts}(\text{speed}) |>
e\_\text{scatter}(\text{dist}) |>
e\_\text{datazoom}()

observe({
  req(input$zoom)
  echarts4rProxy("chart") |>
    e\_\text{dispatch}\_\text{action}_p("dataZoom", startValue = 1, endValue = 10)
})

if (interactive()) {
  shinyApp(ui, server)
}
```
### `e_draft`

**Description**

Add a draft watermark to your graph.

**Usage**

```r
e_draft(e, text = "DRAFT", size = "120px", opacity = 0.4, color = "#d3d3d3")
```

**Arguments**

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `text`: Text to display.
- `size`: Font size of text.
- `opacity, color`: Opacity and color of text.

**Examples**

```r
cars |>
e_charts(speed) |>
e_scatter(dist) |>
e_draft()
```

### `e_draw_p`

**Description**

Draw the chart.

**Usage**

```r
e_draw_p(proxy)
```

**Arguments**

- `proxy`: An `echarts4r` proxy as returned by `echarts4rProxy`.

**Details**

Useful if you set `draw` to `FALSE` in `e_charts`.
Examples

## Not run:
library(shiny)

ui <- fluidPage(
  echarts4rOutput("chart"),
  actionButton("draw", "draw")
)

server <- function(input, output) {
  output$chart <- renderEcharts4r({
    mtcars |>
      e_charts(mpg, draw = FALSE) |>
      e_scatter(qsec) |>
      e_datazoom()
  })

  observeEvent(input$draw, {
    echarts4rProxy("chart") |>
      e_draw_p()
  })
}

if (interactive()) {
  shinyApp(ui, server)
}

## End(Not run)

---

e_error_bar  

### Error bar

**Description**

Add error bars.

**Usage**

e_error_bar(
  e,
  lower,
  upper,
  name = NULL,
  legend = FALSE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
Arguments

e        An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
lower, upper  Lower and upper error bands.
name      name of the serie.
legend    Whether to add serie to legend.
x_index, y_index  Indexes of x and y axis.
coord_system Coordinate system to plot against.
...       Any other option to pass, check See Also section.
itemStyle  mostly used for borderWidth, default 1.5
renderer    name of render function from renderers.js

Examples

df <- data.frame(
  x = factor(c(1, 2)),
  y = c(1, 5),
  upper = c(1.1, 5.3),
  lower = c(0.8, 4.6)
)

df |>  
  e_charts(x) |>
  e_bar(y) |>
  e_error_bar(lower, upper)

# timeline

df <- data.frame(
  x = factor(c(1, 1, 2, 2)),
  y = c(1, 5, 3, 4),
  step = factor(c(1, 2, 1, 2)),
  upper = c(1.1, 5.3),
  lower = c(0.8, 4.6)
)
upper = c(1.1, 5.3, 3.3, 4.2),
lower = c(0.8, 4.6, 2.4, 3.6)
)
df |>  
group_by(step) |>  
e_charts(x, timeline = TRUE) |>  
e_bar(y) |>  
e_error_bar(lower, upper)

---

**e_execute**

**Send**

**Description**

Send new series to chart.

**Usage**

e_execute(proxy)
e_execute_p(proxy)

**Arguments**

- **proxy**
  
  An echarts4r proxy as returned by `echarts4rProxy`.

---

**e_facet**

**Facet**

**Description**

Create facets for multiple plots.

**Usage**

e_facet(
e,  
  rows = NULL,  
  cols = NULL,  
  legend_pos = "top",  
  legend_space = 10,  
  margin_trbl = c(t = 2, r = 2, b = 5, l = 2),  
  h_panel_space = NULL,  
  v_panel_space = NULL
)
e_flip_coords

**Arguments**

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **rows, cols**: Number of rows and columns. If both are `NULL` the number of rows and columns will be determined automatically.
- **legend_pos**: Position of the legend. One of "top", "right", "bottom", "left". Determines to which side the `legend_space` argument applies.
- **legend_space**: Space between legend and plot area. The entered number will be used as percentage.
- **margin_trbl**: Adjusts the size of the outside margin around the plotting area. Default is `c(t = 2, r = 2, b = 5, l = 2)`.
  Numbers are used as percentage of total plotting area. To change only e.g. two sides `c("r" = 8, "l" = 8)` could be used, other sides will use defaults.
- **h_panel_space, v_panel_space**: Horizontal and vertical spacing between the individual grid elements. Expects numeric input, which will be used as percentage of total plotting area. Default ‘NULL’ will automatically add some panel spacing for low dimensional grids.

**Details**

Each serie, i.e.: `e_bar` will be plotted against a facet.

**Examples**

```r
group_size <- 20
n_groups <- 13
df <- data.frame("day" = rep(1:group_size, times=n_groups),
  "temperature" = runif(group_size * n_groups, 10, 40),
  "location" = rep(LETTERS[1:n_groups], each=group_size))

df |>
  group_by(location) |>
  e_charts(day) |>
  e_line(temperature) |>
  e_facet(rows = 4, cols=4, legend_pos = "top", legend_space = 12)
```

---

**Description**

Flip cartesian 2D coordinates.

**Usage**

`e_flip_coords(e)`
Arguments

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.

Examples

```r
df <- data.frame(
  x = LETTERS[1:5],
  y = runif(5, 1, 5),
  z = runif(5, 3, 10)
)

df |> e_charts(x) |> e_bar(y) |> e_line(z) -> plot

plot # normal
e_flip_coords(plot) # flip
```

---

**e_flow_gl**

*Flow GL*

**Description**

Flow GL

**Usage**

```r
e_flow_gl(
  e,
  y,
  sx,
  sy,
  color,
  name = NULL,
  coord_system = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_flow_gl_(
  e,
  y,
  sx,
  sy,
  color = NULL,
  name = NULL,
```
coord_system = NULL,
rm_x = TRUE,
rm_y = TRUE,
... )

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
y Vector position on the y axis.
sx, sy Velocity in respective axis.
color Vector color.
name name of the serie.
coord_system Coordinate system to use.
rm_x, rm_y Whether to remove x and y axis, only applies if coord_system is not null.
... Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

# coordinates
vectors <- expand.grid(0:9, 0:9)
names(vectors) <- c("x", "y")
vectors$sx <- rnorm(100)
vectors$sy <- rnorm(100)
vectors$color <- log10(runif(100, 1, 10))

vectors |> e_charts(x) |> e_flow_gl(y, sx, sy, color) |> e_visual_map(
  min = 0,
  max = 1,
  # log 10
  dimension = 4,
  # x = 0, y = 1, sx = 3, sy = 4
  show = FALSE,
  # hide
  inRange = list(
    color = c(
      "#313695",
      "#4575b4",
      "#74add1",
      "#abd9e9",
      "#e0f3f8",
      "#ffffff",
    )
  )
)
# map
latlong <- seq(-180, 180, by = 5)
wind <- expand.grid(lng = latlong, lat = latlong)
wind$slng <- rnorm(nrow(wind), 0, 200)
wind$slat <- rnorm(nrow(wind), 0, 200)
wind$color <- abs(wind$slat) - abs(wind$slng)
rng <- range(wind$color)
trans <- list(opacity = 0.5) # transparency

wind |>
  e_charts(lng, backgroundColor = "#333") |>
  e_geo() |>
  e_flow_gl(
    lat,
    slng,
    slat,
    color,
    itemStyle = trans,
    particleSize = 2
  ) |>
  e_visual_map(
    color,
    # range
dimension = 4,
    # lng = 0, lat = 1, slng = 2, slat = 3, color = 4
    show = FALSE,
    # hide
    inRange = list(
      color = c(
        "#313695",
        "#4575b4",
        "#74add1",
        "#abd9e9",
        "#e0f3f8",
        "#fffbf7",
        "#fee090",
        "#fd4e61",
        "#f46d43",
        "#d73027",
        "#a50026"
      )
    )
  ) |>
  e_x_axis(
    splitLine = list(show = FALSE)
  ) |>
  e_y_axis(
    splitLine = list(show = FALSE)
  )
e_focus_adjacency_p

Node Adjacency

Description

Focus or unfocus on node adjacency.

Usage

e_focus_adjacency_p(proxy, index, ...)

e_unfocus_adjacency_p(proxy, ...)

Arguments

proxy An echarts4r proxy as returned by `echarts4rProxy`.
index One or more node index to focus on.
... Any other options, see official documentation and details.

Details

Must pass `seriesId`, `seriesIndex`, or `seriesName`, generally `seriesIndex = 0` will work.

Examples

```r
value <- rnorm(10, 10, 2)

nodes <- data.frame(
  name = sample(LETTERS, 10),
  value = value,
  size = value,
  grp = rep(c("grp1", "grp2"), 5),
  stringsAsFactors = FALSE
)

edges <- data.frame(
  source = sample(nodes$name, 20, replace = TRUE),
  target = sample(nodes$name, 20, replace = TRUE),
  stringsAsFactors = FALSE
)
```
## Not run:

```r
library(shiny)

ui <- fluidPage(
    fluidRow(
        column(2,
            numericInput("index", "Node", value = 3, min = 1, max = 9)
        ),
        column(2,
            br(),
            actionButton("focus", "Focus")
        ),
        column(2,
            br(),
            actionButton("unfocus", "Unfocus")
        ),
        fluidRow(
            column(12, echarts4rOutput("graph"))
        )
    ),
    server <- function(input, output, session) {
        output$graph <- renderEcharts4r({
            e_charts() |>
                e_graph() |>
                    e_graph_nodes(nodes, name, value, size, grp) |>
                        e_graph_edges(edges, source, target)
        })
        observeEvent(input$focus, {
            echarts4rProxy("graph") |>
                e_focus_adjacency_p(
                    seriesIndex = 0,
                    index = input$index
                )
        })
        observeEvent(input$unfocus, {
            echarts4rProxy("graph") |>
                e_unfocus_adjacency_p(seriesIndex = 0)
        })
    }
    if (interactive()) {
        shinyApp(ui, server)
    }
    ## End(Not run)
```
### e_format_axis

**Formatters**

#### Description

Simple formatters as helpers.

#### Usage

- `e_format_axis(e, axis = "y", suffix = NULL, prefix = NULL, ...)`
- `e_format_x_axis(e, suffix = NULL, prefix = NULL, ...)`
- `e_format_y_axis(e, suffix = NULL, prefix = NULL, ...)`

#### Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **axis**: Axis to apply formatter to.
- **suffix, prefix**: Suffix and prefix of label.
- **...**: Any other arguments to pass to `e_axis`.

#### Examples

```r
# Y = %
df <- data.frame(
  x = 1:10,
  y = round(runif(10, 1, 100), 2)
)

df |> e_charts(x) |> e_line(y) |> e_format_y_axis(suffix = ")") |> e_format_x_axis(prefix = "A")```
Description

Add a funnel.

Usage

e_funnel(e, values, labels, name = NULL, legend = TRUE, rm_x = TRUE, rm_y = TRUE, ...)
e_funnel_(e, values, labels, name = NULL, legend = TRUE, rm_x = TRUE, rm_y = TRUE, ...)

Arguments

- **e**: An echarts4r object as returned by `e_charts`.
- **values, labels**: Values and labels of funnel.
- **name**: Name of the serie.
- **legend**: Whether to add serie to legend.
- **rm_x, rm_y**: Whether to remove x and y axis, defaults to TRUE.
- **...**: Any other option to pass to bar or line char types.

Details

No bind argument here, with a funnel bind = labels.

See Also

- Additional arguments
Examples

```r
funnel <- data.frame(
  stage = c("View", "Click", "Purchase"),
  value = c(80, 30, 20)
)

funnel |> e_charts() |> e_funnel(value, stage)
```

---

**e_gauge**

**Description**

Plot a gauge.

**Usage**

```r
e_gauge(e, value, name, rm_x = TRUE, rm_y = TRUE, ...)
e_gauge_(e, value, name, rm_x = TRUE, rm_y = TRUE, ...)
```

**Arguments**

- `e` An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `value` Value to gauge.
- `name` Text on gauge.
- `rm_x`, `rm_y` Whether to remove x and y axis, defaults to TRUE.
- `...` Any other option to pass, check See Also section.

**See Also**

Additional arguments

**Examples**

```r
e_charts() |> e_gauge(57, "PERCENT")

# timeline
data.frame(time = 2015:2017) |> group_by(time) |> e_charts(timeline = TRUE) |> e_gauge(
c(57, 23, 65),
c("percent", "percentage", "cases")
)
```
Description

Initialise geo.

Usage

e_geo(e, map = "world", ..., rm_x = TRUE, rm_y = TRUE)

Arguments

e              An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
map            Map type.
...            Any other option to pass, check See Also section.
rm_x, rm_y     Whether to remove x and y axis, defaults to TRUE.

See Also

Additional arguments

Examples

flights <- read.csv(
  paste0(
    "https://raw.githubusercontent.com/plotly/datasets/",
    "master/2011_february_aa_flight_paths.csv"
  )
)

flights |>  
e_charts() |>  
e_geo() |>  
e_lines(  
  start_lon,
  start_lat,
  end_lon,
  end_lat,
  name = "flights",
  lineStyle = list(normal = list(curveness = 0.3))
)
Description

Initialise geo 3D.

Usage

```r
e_geo_3d(e, serie, color, type = "world", rm_x = TRUE, rm_y = TRUE, ...)
```

e_geo_3d_(
  e,
  serie = NULL,
  color = NULL,
  type = "world",
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

Arguments

- `e` An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `serie` Column name of serie to plot.
- `color` Color.
- `type` Map type.
- `rm_x, rm_y` Whether to remove x and y axis, defaults to TRUE.
- `...` Any other option to pass, check See Also section.

See Also

e_country_names, Additional arguments

Examples

```r
choropleth <- data.frame(
  countries = c(
    "France",
    "Brazil",
    "China",
    "Russia",
    "Canada",
    "India",
    "United States",
    "Argentina",
    "Australia"
  ),
  values = c(1, 2, 3, 4, 5, 6, 7, 8, 9)
)
```

```r
library(echarts4r)

e_e_charts(choropleth, 
  e_geo_3d(e, serie = "countries", color = "red", type = "world", rm_x = TRUE, rm_y = TRUE),
  e_charts_options(
    title = "World Map with Countries",
    map_type = "world",
    highlight_color = "#FF0000",
    emphasis_color = "#00FF00",
    ...))
```
e_get_data

Get data

Description
Get data passed to e_charts.

Usage
e_get_data(e)

Arguments
e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.

Value
A list of data.frames, one for each group.

Examples
echart <- cars |> 
  e_charts(speed) |> 
  e_scatter(dist) |> 
  e_lm(dist ~ speed)

  echart

  e_get_data(echart)[[1]]
**e_get_zr**

**Blank Area**

*Description*

Use this function to capture a click on a blank area of the canvas. Note that this may stop other "click" events from working.

*Usage*

```r
e_get_zr()
```

---

**e_globe**

**Globe**

*Description*

Add globe.

*Usage*

```r
e_globe(e, environment = NULL, base_texture = NULL, height_texture = NULL, ...)
```

*Arguments*

- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `environment` Texture of background.
- `base_texture` Base texture of globe.
- `height_texture` Texture of height.
- `...` Any other option to pass, check See Also section.

*See Also*

`e_country_names`, Additional arguments

*Examples*

```r
## Not run:
url <- paste0(
  "https://echarts.apache.org/examples/",
  "data-gl/asset/data/population.json"
)
data <- jsonlite::fromJSON(url)
data <- as.data.frame(data)
names(data) <- c("lon", "lat", "value")
```
```r
data |> 
e_charts(lon) |> 
e_globe(
  displacementScale = 0.04
) |> 
e_bar_3d(lat, value, "globe") |> 
e_visual_map(show = FALSE)
## End(Not run)
```

---

**Description**

Create a graph.

**Usage**

```r
e_graph(e, layout = "force", name = NULL, rm_x = TRUE, rm_y = TRUE, ...)
e_graph_gl(  
e,  
layout = "force",  
name = NULL,  
rm_x = TRUE,  
rm_y = TRUE,  
...,  
itemStyle = list(opacity = 1)
)
e_graph_nodes(  
e,  
nodes,  
names,  
value,  
size,  
category,  
symbol = NULL,  
legend = TRUE
)
e_graph_edges(e, edges, source, target, value, size)
```
Arguments

- `e`: An echarts4 object as returned by `e_charts`.
- `layout`: Layout, one of 'force', 'none' or 'circular'.
- `name`: Name of graph.
- `rm_x, rm_y`: Whether to remove the x and y axis, defaults to TRUE.
- `...`: Any other parameter.
- `itemStyle`: This option is available for for GL and canvas graph but is only necessary for GL.
- `nodes`: Data.frame of nodes.
- `names`: Names of nodes, unique.
- `value`: Values of nodes or edges.
- `size`: Sizes of nodes or edges.
- `category`: Group of nodes (i.e.: group membership).
- `symbol`: Symbols of nodes.
- `legend`: Whether to add serie to legend.
- `edges`: Data.frame of edges.
- `source, target`: Column names of source and target.

See Also

- Additional arguments, e_modularity

Examples

```r
value <- rnorm(10, 10, 2)

nodes <- data.frame(
  name = sample(LETTERS, 10),
  value = value,
  size = value,
  symbol = sample(c("circle", "rect", "triangle"), 10, replace = TRUE),
  grp = rep(c("grp1", "grp2"), 5),
  stringsAsFactors = FALSE
)

value_edges <- sample(1:100, 20, replace = TRUE)
edges <- data.frame(
  source = sample(nodes$name, 20, replace = TRUE),
  target = sample(nodes$name, 20, replace = TRUE),
  value = value_edges,
  size = ceiling(value_edges/20),
  stringsAsFactors = FALSE
)

e_charts() |> 
e_graph() |>
```
# Use graphGL for larger networks

```r
define nodes <- data.frame(
  name = paste0(LETTERS, 1:1000),
  value = rnorm(1000, 10, 2),
  size = rnorm(1000, 10, 2),
  grp = rep(c("grp1", "grp2"), 500),
  stringsAsFactors = FALSE)

define edges <- data.frame(
  source = sample(nodes$name, 2000, replace = TRUE),
  target = sample(nodes$name, 2000, replace = TRUE),
  stringsAsFactors = FALSE
)

e_charts() |>  
e_graph_gl() |>  
e_graph_nodes(nodes, name, value, size, grp) |>  
e_graph_edges(edges, source, target)
```

---

**e_graphic_g**

**Graphic**

**Description**

Low level API to define graphic elements.

**Usage**

- `e_graphic_g(e, ...)`
- `e_group_g(e, ...)`
- `e_image_g(e, ...)`
- `e_text_g(e, ...)`
- `e_rect_g(e, ...)`
- `e_circle_g(e, ...)`
- `e_ring_g(e, ...)`
- `e_sector_g(e, ...)`
\texttt{e_graphic\_g}

\begin{verbatim}
e\_arc\_g(e, ...)
e\_polygon\_g(e, ...)
e\_polyline\_g(e, ...)
e\_line\_g(e, ...)
e\_bezier\_curve\_g(e, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{e} \hspace{1cm} An \texttt{echarts4r} object as returned by \texttt{e\_charts} or a proxy as returned by \texttt{echarts4rProxy}.
  \item \texttt{...} \hspace{1cm} Any other option to pass, check See Also section.
\end{itemize}

Functions

\begin{itemize}
  \item \texttt{e\_graphic\_g} to initialise graphics, entirely optional.
  \item \texttt{e\_group\_g} to create group, the children of which will share attributes.
  \item \texttt{e\_image\_g} to a png or jpg image.
  \item \texttt{e\_text\_g} to add text.
  \item \texttt{e\_rect\_g} to add a rectangle.
  \item \texttt{e\_circle\_g} to add a circle.
  \item \texttt{e\_ring\_g} to add a ring.
  \item \texttt{e\_sector\_g}
  \item \texttt{e\_arc\_g} to create an arc.
  \item \texttt{e\_polygon\_g} to create a polygon.
  \item \texttt{e\_polyline\_g} to create a polyline.
  \item \texttt{e\_line\_g} to draw a line.
  \item \texttt{e\_bezier\_curve\_g} to draw a quadratic bezier curve or cubic bezier curve.
\end{itemize}

Note

Some elements, i.e.: \texttt{e\_image\_g} may not display in the RStudio browser but will work fine in your browser, R markdown documents and Shiny applications.

See Also

\begin{verbatim}
official documentation
\end{verbatim}
Examples

# may not work in RStudio viewer
# Open in browser
cars |> 
e_charts(speed) |> 
e_scatter(dist) |> 
e_image_g( 
  right = 20, 
  top = 20, 
  z = -999, 
  style = list( 
    image = "https://www.r-project.org/logo/Rlogo.png", 
    width = 150, 
    height = 150, 
    opacity = .6 
  ) 
)

Description

Customise grid.

Usage

e_grid(e, index = NULL, ...)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echart4rProxy.

index Index of axis to customise.

... Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

USA Arrests |> 
e_charts(UrbanPop) |> 
e_line(Assault, smooth = TRUE) |> 
e_area(Murder, y.index = 1, x.index = 1) |> 
e_y_axis(gridIndex = 1) |> 
e_x_axis(gridIndex = 1) |> 
e_grid(height = "40%") |> 
e_grid(height = "40", top = "55%")
Description

Customise grid.

Usage

e_grid_3d(e, index = 0, ...)

Arguments

e
An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.

index
Index of axis to customise.

...
Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

# phony data
v <- LETTERS[1:10]
matrix <- data.frame(  
x = sample(v, 300, replace = TRUE),
y = sample(v, 300, replace = TRUE),
z1 = rnorm(300, 10, 1),
z2 = rnorm(300, 10, 1),
stringsAsFactors = FALSE)
  ) |> dplyr::group_by(x, y) |> dplyr::summarise(  
z1 = sum(z1),
z2 = sum(z2)
  ) |> dplyr::ungroup()

trans <- list(opacity = 0.4) # transparency
emphasis <- list(itemStyle = list(color = "#313695"))

matrix |> e_charts(x) |> e_bar_3d(y, z1, stack = "stack", name = "Serie 1", itemStyle = trans, emphasis = emphasis) |> e_bar_3d(y, z2, stack = "stack", name = "Serie 2", itemStyle = trans, emphasis = emphasis) |> e_grid_3d(splitLine = list(lineStyle = list(color = "blue")))
**e_heatmap**

**Heatmap**

---

**Description**

Draw heatmap by coordinates.

**Usage**

```r
e_heatmap(
  e,
  y,
  z,
  bind,
  name = NULL,
  coord_system = "cartesian2d",
  rm_x = TRUE,
  rm_y = TRUE,
  calendar = NULL,
  ...
)
```

```r
e_heatmap_(
  e,
  y,
  z = NULL,
  bind = NULL,
  name = NULL,
  coord_system = "cartesian2d",
  rm_x = TRUE,
  rm_y = TRUE,
  calendar = NULL,
  ...
)
```

**Arguments**

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **y, z**: Coordinates and values.
- **bind**: Binding between datasets, namely for use of `e_brush`.
- **name**: Name of the series.
- **coord_system**: Coordinate system to plot against, takes `cartesian2d`, `geo` or `calendar`.
- **rm_x, rm_y**: Whether to remove x and y axis, only applies if `coord_system` is not set to `cartesian2d`.
- **calendar**: The index of the calendar to plot against.
- **...**: Any other option to pass, check See Also section.
See Also

Additional arguments

Examples

```r
v <- LETTERS[1:10]
matrix <- data.frame(
  x = sample(v, 300, replace = TRUE),
  y = sample(v, 300, replace = TRUE),
  z = rnorm(300, 10, 1),
  stringsAsFactors = FALSE
) |
  dplyr::group_by(x, y) |
  dplyr::summarise(z = sum(z)) |
  dplyr::ungroup()
matrix |
  e_charts(x) |
  e_heatmap(y, z, itemStyle = list(emphasis = list(shadowBlur = 10))) |
  e_visual_map(z)

# calendar
dates <- seq.Date(as.Date("2017-01-01"), as.Date("2018-12-31"), by = "day")
values <- rnorm(length(dates), 20, 6)
year <- data.frame(date = dates, values = values)

year |
  e_charts(date) |
  e_calendar(range = "2018") |
  e_heatmap(values, coord_system = "calendar") |
  e_visual_map(max = 30)

# calendar multiple years
year |
  dplyr::mutate(year = format(date, "%Y")) |
  group_by(year) |
  e_charts(date) |
  e_calendar(range = "2017", top = 40) |
  e_calendar(range = "2018", top = 260) |
  e_heatmap(values, coord_system = "calendar") |
  e_visual_map(max = 30)

# map
quakes |
  e_charts(long) |
  e_geo(
    boundingCoords = list(
      c(190, -10),
      c(180, -40)
    )
  ) |
```
e_hide_grid_lines

`e_hide_grid_lines(e, which = c("x", "y"))`

**Description**

A convenience function to easily hide grid lines.

**Usage**

```r
e_hide_grid_lines(e, which = c("x", "y"))
```

**Arguments**

- `e`  
  An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.

- `which`  
  Which axis grid lines to hide.

**Examples**

```r
cars |>
e_charts(speed) |>
e_scatter(dist) |>
e_hide_grid_lines()
```
Description

Proxies to highlight and downplay series.

Usage

e_highlight_p(proxy, series_index = NULL, series_name = NULL)
e_downplay_p(proxy, series_index = NULL, series_name = NULL)

Arguments

proxy An echarts4r proxy as returned by echarts4rProxy.
series_index Series index, can be a vector.
series_name Series Name, can be vector.

Examples

## Not run:
library(shiny)

ui <- fluidPage(
  fluidRow(
    column(3,
      actionButton("highlightmpg", "Highlight MPG")
    ),
    column(3,
      actionButton("highlighthp", "Highlight HP")
    ),
    column(3,
      actionButton("downplaympg", "Downplay MPG")
    ),
    column(3,
      actionButton("downplayhp", "Downplay HP")
    ),
    echarts4rOutput("plot")
  )
)

server <- function(input, output, session) {
  output$plot <- renderEcharts4r({
    mtcars |>
e_histogram

```r
e_charts(mpg) |> 
e_line(disp) |> 
e_line(hp, name = "HP") # explicitly pass name
})

# highlight
observeEvent(input$highlightmpg, { 
echarts4rProxy("plot") |> 
e_highlight_p(series_index = 0) # using index
})

observeEvent(input$highlighthp, { 
echarts4rProxy("plot") |> 
e_highlight_p(series_name = "HP") # using name
})

# downplay
observeEvent(input$downplaympg, { 
echarts4rProxy("plot") |> 
e_downplay_p(series_name = "disp")
})

observeEvent(input$downplayhp, { 
echarts4rProxy("plot") |> 
e_downplay_p(series_index = 1)
})

if (interactive()) {
  shinyApp(ui, server)
}

## End(Not run)
```

---

**e_histogram**

### Histogram & Density

**Description**

Add a histogram or density plots.

**Usage**

```r
e_histogram(
e, 
serie, 
breaks = "Sturges",
```
Arguments

- **e**: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **serie**: Column name of serie to plot.
breaks                  Passed to `hist`.
name                   name of the serie.
legend                 Whether to add serie to legend.
bar_width              Width of bars.
x_index, y_index       Indexes of x and y axis.
...                    Any other option to pass, check See Also section.
smooth                 Whether to use smoothed lines, passed to `e_line`.

See Also

Additional arguments for histogram, Additional arguments for density

Examples

```
mtcars |>
e_charts() |>
e_histogram(mpg, name = "histogram") |>
e_density(mpg, areaStyle = list(opacity = .4), smooth = TRUE, name = "density", y_index = 1) |>
e_tooltip(trigger = "axis")
```

```
# timeline
mtcars |>
  group_by(cyl) |>
e_charts(timeline = TRUE) |>
e_histogram(mpg, name = "histogram") |>
e_density(mpg, name = "density", y_index = 1)
```

---

**e_inspect**  
*To & From JSON*

**Description**

Get JSON options from an echarts4r object and build one from JSON.

**Usage**

```
e_inspect(e, json = FALSE, ...)  
echarts_from_json(txt)
```

**Arguments**

- **e**  
  An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **json**  
  Whether to return the JSON, otherwise returns a list.
- **...**  
  Additional options to pass to `toJSON`.
- **txt**  
  JSON character string, url, or file.
Details

txt should contain the full list of options required to build a chart. This is subsequently passed to the `setOption` ECharts (JavaScript) function.

Value

e.inspect Returns a list if `json` is FALSE and a JSON string otherwise. `echarts_from_json` returns an object of class `echarts4r`.

Note

Must be passed as last option.

Examples

```r
p <- cars |> e_charts(dist) |> e_scatter(speed, symbol_size = 10)
p # plot

# extract the JSON
json <- p |> e_inspect(
    json = TRUE,
    pretty = TRUE
)

# print json
json

# rebuild plot
echarts_from_json(json) |> e_theme("dark") # modify
```

---

**e_labels**

*Format labels*

**Description**

Format labels

**Usage**

e_labels(e, show = TRUE, position = "top", ...)

Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **show**: Set to `TRUE` to show the labels.
- **position**: Position of labels, see official documentation for the full list of options.
- **...**: Any other options see documentation for other options.

Examples

```r
mtcars |> 
  e_chart(wt) |> 
  e_scatter(qsec, cyl) |> 
  e_labels(fontSize = 9)

mtcars |> 
  group_by(cyl) |> 
  e_chart(wt) |> 
  e_scatter(qsec, mpg) |> 
  e_labels(fontSize = 9)

# timeline
mtcars |> 
  group_by(cyl) |> 
  e_chart(wt) |> 
  e_scatter(qsec, mpg) |> 
  e_labels(fontSize = 9)
```

---

e_leaflet

**Leaflet**

**Description**

Leaflet extension.

**Usage**

```r
e_leaflet(e, roam = TRUE, ...)
e_leaflet_tile(
  e,
  template = "https://{s}.tile.openstreetmap.fr/hot/{z}/{x}/{y}.png",
  options = NULL,
  ...
)
```
**Arguments**

- **e**
  An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.

- **roam**
  Whether to allow the user to roam.

- **...**
  Any other option to pass, check See Also section.

- **template**
  u1Template, should not be changed.

- **options**
  List of options, including attribution and label.

**Note**

Will not render in the RStudio, open in browser.

**Examples**

```r
## Not run:
url <- paste0(
  "https://echarts.apache.org/examples/",
  "data-gl/asset/data/population.json"
)
data <- jsonlite::fromJSON(url)
data <- as.data.frame(data)
names(data) <- c("lon", "lat", "value")
data$value <- log(data$value)

data |>
e_charts(lon) |>
e_leaflet() |>
e_leaflet_tile() |>
e_scatter(lat, size = value, coord_system = "leaflet")
## End(Not run)
```

---

**e_legend**  
**Legend**

**Description**

Customise the legend.

**Usage**

```r
e_legend(e, show = TRUE, type = c("plain", "scroll"), icons = NULL, ...)
```
Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **show**: Set to FALSE to hide the legend.
- **type**: Type of legend, plain or scroll.
- **icons**: A optional list of icons the same length as there are series, see example.
- **...**: Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

```r
# with legend
e <- cars |> e_charts(speed) |> e_scatter(dist, symbol_size = 5)

e

# without legend
e |> e_legend(show = FALSE)

e |
    e_legend(icons = list(path)

    )
```

---

e_line  

**Line**

Description

Add line serie.
Usage

e_line(
  e,
  serie,
  bind,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)

e_line_(
  e,
  serie,
  bind = NULL,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)

Arguments

e           An echart4r object as returned by e_charts or a proxy as returned by echart4rProxy.
serie       Column name of serie to plot.
bind        Binding between datasets, namely for use of e_brush.
name        name of the serie.
legend      Whether to add serie to legend.
x_index, y_index
            Indexes of x and y axis.
coord_system Coordinate system to plot against.
...          Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

iris |>
  group_by(Species) |>
  e_charts(Sepal.Length) |>
  e_line(Sepal.Width) |>
# timeline
iris |>  
  group_by(Species) |>  
  e_charts(Sepal.Length, timeline = TRUE) |>  
  e_line(Sepal.Width) |>  
  e_tooltip(trigger = "axis")

---

## e_lines

**Lines**

**Description**

Add lines.

**Usage**

```r
e_lines(
  e,
  source_lon,  
  source_lat,
  target_lon,
  target_lat,
  source_name,
  target_name,
  value,
  coord_system = "geo",
  name = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
```

```r
e_lines_(
  e,
  source_lon,  
  source_lat,
  target_lon,
  target_lat,
  source_name = NULL,
  target_name = NULL,
  value = NULL,
  coord_system = "geo",
  name = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
```
Arguments

e An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
source_lon, source_lat, target_lon, target_lat
coordinates.
source_name, target_name
Names of source and target.
value Value of edges.
coord_system Coordinate system to use, one of `geo`, or `cartesian2d`.
name name of the serie.
rm_x, rm_y Whether to remove x and y axis, defaults to TRUE.
... Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

```r
flights <- read.csv(
paste0(
  "https://raw.githubusercontent.com/plotly/datasets/",
  "master/2011_february_aa_flight_paths.csv"
)
)

flights |>
  e_charts() |>
  e_geo() |>
  e_lines(
    start_lon,
    start_lat,
    end_lon,
    end_lat,
    airport1,
    airport2,
    cnt,
    name = "flights",
    lineStyle = list(normal = list(curveness = 0.3))
  ) |>
  e_tooltip(
    trigger = "item",
    formatter = htmlwidgets::JS("function(params){
      return(
        params.seriesName + '<br />' +
        params.data.source_name + ' -> ' +
        params.data.target_name + ':' + params.value
      )
    }
  )
```
```
flights $grp <- rep(LETTERS[1:2], 89)
flights |> group_by(grp) |> e_charts(timeline = TRUE) |> e_geo() |> e_lines(
  start_lon,
  start_lat,
  end_lon,
  end_lat,
  cnt,
  coord_system = "geo"
)
```

---

**Description**

Add 3D lines.

**Usage**

```r
e_lines_3d(
  e,
  source_lon,
  source_lat,
  target_lon,
  target_lat,
  source_name,
  target_name,
  value,
  name = NULL,
  coord_system = "globe",
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
```

```r
e_line_3d(
  e,
  y,
  z,
```
name = NULL,
coord_system = NULL,
rm_x = TRUE,
rm_y = TRUE,
... )

e_lines_3d_(
e,
source_lon,
source_lat,
target_lon,
target_lat,
source_name = NULL,
target_name = NULL,
value = NULL,
name = NULL,
coord_system = "globe",
rm_x = TRUE,
rm_y = TRUE,
...
)

e_line_3d_(
e,
y,
z,
name = NULL,
coord_system = NULL,
rm_x = TRUE,
rm_y = TRUE,
...
)

Arguments

- **e** An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **source_lon, source_lat, target_lon, target_lat** Coordinates.
- **source_name, target_name** Names of source and target.
- **value** Value of edges.
- **name** Name of the serie.
- **coord_system** Coordinate system to use, such as cartesian3D, or globe.
- **rm_x, rm_y** Whether to remove x and y axis, defaults to TRUE.
- **...** Any other option to pass, check See Also section.
- **y, z** Coordinates of lines.
See Also

Additional arguments for lines 3D, Additional arguments for line 3D
https://echarts4r-assets.john-coene.com

Examples

```r
# get data
flights <- read.csv(
  paste0(
    "https://raw.githubusercontent.com/plotly/datasets/",
    "master/2011_february_aa_flight_paths.csv"
  )
)

# Lines 3D
# Globe
flights |>
  e_charts() |>
  e_globe(
    displacementScale = 0.05
  ) |>
  e_lines_3d(
    start_lon,
    start_lat,
    end_lon,
    end_lat,
    name = "flights",
    effect = list(show = TRUE)
  ) |>
  e_legend(FALSE)

# Geo 3D
flights |>
  e_charts() |>
  e_geo_3d() |>
  e_lines_3d(
    start_lon,
    start_lat,
    end_lon,
    end_lat,
    coord_system = "geo3D"
  )

# groups
flights$grp <- rep(LETTERS[1:2], 89)

flights |>
  group_by(grp) |>
  e_charts() |>
  e_geo_3d() |>
  e_lines_3d(
```
start_lon,
start_lat,
end_lon,
end_lat,
coord_system = "geo3D"
)

# line 3D
df <- data.frame(
x = 1:100,
y = runif(100, 10, 25),
z = rnorm(100, 100, 50)
)

df |> |
e_charts(x) |> |
e_line_3d(y, z) |> |
e_visual_map() |> |
e_title("nonsense")

# timeline
df$grp <- rep(LETTERS[1:5], 20)

df |> |
group_by(grp) |> |
e_charts(x) |> |
e_line_3d(y, z) |> |
e_visual_map() |> |
e_title("nonsense")

---

e_lines_gl

### Lines WebGL

#### Description

Draw WebGL lines.

#### Usage

```r
e_lines_gl(e, data, coord_system = "geo", ...)
```

#### Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **data**: A list.
- **coord_system**: Coordinate system to plot against.
- **...**: Any other options (this series type is mostly undocumented).
Description

Draw liquid fill.

Usage

e_liquid(e, serie, color, rm_x = TRUE, rm_y = TRUE, ...)
e_liquid_(e, serie, color = NULL, rm_x = TRUE, rm_y = TRUE, ...)

Arguments

e: An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie: Column name of serie to plot.
color: Color to plot.
rm_x, rm_y: Whether to remove x and y axis, defaults to TRUE.
...: Any other option to pass, check See Also section.

See Also

official documentation

Examples

df <- data.frame(val = c(0.6, 0.5, 0.4))

df |> e_charts() |> e_liquid(val) |> e_theme("dark")

Description

simply pass a list of options, similar to a JSON.

Usage

e_list(e, list, append = FALSE)
Arguments

e An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
list A list of options passed to `setOptions`.
append if TRUE the list is appended to the options, otherwise it overwrites everything.

Examples

```r
N <- 20 # data points

opts <- list(
  xAxis = list(
    type = "category",
    data = LETTERS[1:N]
  ),
  yAxis = list(
    type = "value"
  ),
  series = list(
    list(
      type = "line",
      data = round(runif(N, 5, 20))
    )
  )
)

e_charts() |> 
  e_list(opts)
```

---

**e_lm** Smooth

Description

Plot formulas.

Usage

```r
e_lm(
  e, 
  formula, 
  name = NULL, 
  legend = TRUE, 
  symbol = "none", 
  smooth = TRUE, 
  model_args = list(), 
  ...
)
```
\texttt{e_lm}

\begin{verbatim}
e_glm(
e, formula,
name = NULL,
legend = TRUE,
symbol = "none",
smooth = TRUE,
model_args = list(), ...
)
e_loess(
e, formula,
name = NULL,
legend = TRUE,
symbol = "none",
smooth = TRUE,
x_index = 0,
y_index = 0,
model_args = list(), ...
)
\end{verbatim}

Arguments

- \texttt{e} An \texttt{echarts4r} object as returned by \texttt{e_charts} or a proxy as returned by \texttt{echarts4rProxy}.
- \texttt{formula} formula to pass to \texttt{lm}.
- \texttt{name} name of the serie.
- \texttt{legend} Whether to add serie to legend.
- \texttt{symbol} Symbol to use in \texttt{e_line}.
- \texttt{smooth} Whether to smooth the line.
- \texttt{model_args} Arguments to pass to the underlying model.
- \texttt{x_index}, \texttt{y_index} Indexes of x and y axis.

Examples

\begin{verbatim}
iris |>
group_by(Species) |>
e_charts(Sepal.Length) |>
e_scatter(Sepal.Width) |>
e_lm(Sepal.Width ~ Sepal.Length) |>
e_x_axis(min = 4)
mtcars |>
\end{verbatim}
```r
eCharts(disp) |> 
eScatter(mpg, qsec) |> 
loess(mpg ~ disp, smooth = TRUE, showSymbol = FALSE)

# timeline
iris |> 
group_by(Species) |> 
eCharts(Sepal.Length, timeline = TRUE) |> 
eScatter(Sepal.Width) |> 
élm(Sepal.Width ~ Sepal.Length) |> 
ex_axis(min = 4, max = 8) |> 
ey_axis(max = 5)
```

### Description
Change the locale to auto-translate days of the week, etc.

### Usage
```
e_locale(e, locale)
e_locale_manual(e, locale, path)
```

### Arguments
- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `locale` Locale to set to.
- `path` Path to the local file to use.

### Details
The "manual" function expects a file to use for translations. You can browse the `.js` files [here](https://github.com/apache/echarts/tree/master/i18n) to have an idea of what they should look like.

### Locales
- CS - DE - EN - ES - FI - FR - JA - PT (brazil) - SI - TH - ZH

### Examples
```
# top right corner zoom is in
# French
cars |> 
eCharts(speed) |> 
eScatter(dist) |> 
eDatazoom() |>
```

<table>
<thead>
<tr>
<th>e_locale</th>
<th>Locale</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Description

Draw maps.

Usage

\[ e_{\text{map}}(e, \text{serie}, \text{map} = "\text{world}" , \text{name} = \text{NULL}, \text{rm}_x = \text{TRUE}, \text{rm}_y = \text{TRUE}, \ldots) \]

\[ e_{\text{map}}(\text{e}, \text{serie} = \text{NULL}, \text{map} = "\text{world}" , \text{name} = \text{NULL}, \text{rm}_x = \text{TRUE}, \text{rm}_y = \text{TRUE}, \ldots) \]

\[ e_{\text{svg}}(e, \text{serie}, \text{map} = "\text{world}" , \text{name} = \text{NULL}, \text{rm}_x = \text{TRUE}, \text{rm}_y = \text{TRUE}, \ldots) \]

\[ e_{\text{svg}}(\text{e}, \text{serie} = \text{NULL}, \text{map} = "\text{world}" , \text{name} = \text{NULL}, \text{rm}_x = \text{TRUE}, \text{rm}_y = \text{TRUE}, \ldots) \]

\[ e_{\text{map}_3d}(\text{e}, \text{serie}, \text{map} = "\text{world}" , \text{name} = \text{NULL}, \text{coord\_system} = \text{NULL}, \text{rm}_x = \text{TRUE}, \text{rm}_y = \text{TRUE}, \ldots) \]
e_map_3d_(
  e,
  serie = NULL,
  map = "world",
  name = NULL,
  coord_system = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_map_3d_custom(
  e,
  id,
  value,
  height,
  map = NULL,
  name = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie Values to plot.
map Map type.
name name of the serie.
rm_x, rm_y Whether to remove x and y axis, defaults to TRUE.
... Any other option to pass, check See Also section.
coord_system Coordinate system to use, one of cartesian3D, geo3D, globe.
id, value, height Columns corresponding to registered map.

See Also

e_country_names, Additional map arguments, Additional map 3D arguments

Examples

## Not run:
choropleth <- data.frame(
countries = c(
  "France",
  "Brazil",
  "China",
  "Russia",
...
e_map

"Canada",
"India",
"United States",
"Argentina",
"Australia"
),
values = round(runif(9, 10, 25))
)

choropleth |>
  e_charts(countries) |>
  e_map(values) |>
  e_visual_map(min = 10, max = 25)

choropleth |>
  e_charts(countries) |>
  e_map_3d(values, shading = "lambert") |>
  e_visual_map(min = 10, max = 30)

# custom
buildings <- jsonlite::read_json(
paste0(
  "https://echarts.apache.org/examples/",
  "data-gl/asset/data/buildings.json"
)
)

heights <- purrr::map(buildings$features, "properties") |>
  purrr::map("height") |>
  unlist()

names <- purrr::map(buildings$features, "properties") |>
  purrr::map("name") |>
  unlist()

data <- dplyr::tibble(
  name = names,
  value = round(runif(length(names), 0, 1), 6),
  height = heights / 10
)

data |>
  e_charts() |>
  e_map_register("buildings", buildings) |>
  e_map_3d_custom(name, value, height) |>
  e_visual_map(
    show = FALSE,
    min = 0.4,
    max = 1
  )

# timeline
choropleth <- data.frame
countries = rep(choropleth$countries, 3)
)
>
dplyr::mutate(
  grp = c(
    rep(2016, nrow(choropleth)),
    rep(2017, nrow(choropleth)),
    rep(2018, nrow(choropleth))
  ),
  values = runif(27, 1, 10)
)

choropleth |> 
  group_by(grp) |> 
  e_charts(countries, timeline = TRUE) |> 
  e_map(values) |> 
  e_visual_map(min = 1, max = 10)

choropleth |> 
  group_by(grp) |> 
  e_charts(countries, timeline = TRUE) |> 
  e_map_3d(values) |> 
  e_visual_map(min = 1, max = 10)

## End(Not run)

---

e_map_register  Register map

Description

Register a geojson map.

Usage

e_map_register(e, name, json, ...)

e_svg_register(e, name, svg)

e_map_register_p(
  name,
  json,
  async = FALSE,
  session = shiny::getDefaultReactiveDomain()
)

e_map_register_ui(name, json, async = FALSE)
**Arguments**

- **e**: An echarts4r object as returned by `e_charts`.
- **name**: Name of map, to use in `e_map`.
- **json, svg**: Geojson, or SVG.
- **...**: Additional options passed to `registerMap`.
- **async**: Whether to read the file asynchronously.
- **session**: A valid Shiny session.

**Details**

e_map_register_p is not truly a proxy as it does not require a chart to function. While the function `e_map_register_ui` is meant to register the map globally in the Shiny UI, not that then `json` must be accessible from the UI (generally www folder).

**Examples**

```r
## Not run:
json <- jsonlite::read_json("https://echarts.apache.org/examples/data/asset/geo/USA.json")
USArrests |>
  tibble::rownames_to_column("states") |>
  e_charts(states) |>
  e_map_register("USA", json) |>
  e_map(Murder, map = "USA") |>
  e_visual_map(Murder)
## End(Not run)
```

---

**e_mark_p**  
**Mark**

**Description**

Mark points, lines, and areas with a proxy (`echarts4rProxy()`).

**Usage**

```
e_mark_p(e, type, serie_index, data, ...)
e_mark_p_(e, type, serie_index, data = NULL, ...)
```
Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **type**: Type of mark: 'point', 'line' or 'area', defaults to 'point'.
- **serie_index**: Single index of serie to mark on, defaults to 1. Proxy doesn’t know series’ names, so it only uses index.
- **data**: Location of point, line or area, defaults to NULL.
- **...**: Any other option to pass, check See Also section.

Details

Allows the three type of marks to work with `echarts4rProxy()`

Examples

```r
library(shiny)
library(dplyr)

ui <- fluidPage(
  fluidRow(
    column(3, actionButton("pxy", "Marks")),
    column(3, checkboxInput("tln", "Timeline", value = FALSE)
  ),
  echarts4rOutput("plot")
)

server <- function(input, output) {
  data(EuStockMarkets)
  bb <- as.data.frame(EuStockMarkets) |>
    slice_head(n = 150) |>
    mutate(day = 1:n())
  output$plot <- renderEcharts4r({
    react()
  })

  observeEvent(input$pxy, {
    echarts4rProxy("plot", data = NULL) |> 
      e_mark_p(
        type = "line",
        serie_index = 1,
        data = list(type = "average"),
        lineStyle = list(type = "dashed", color = "cyan")
      ) |> 
      e_mark_p(
        serie_index = 2,
        data = list(
          type = "area",
          style = list(color = "red")
        )
      )
  })
}
```
e_mark_point

Mark point

```r
e_mark_point
  xAxis = bb$day[60],
yAxis = bb$SMI[60],
  value = "pnt"
)
)e_mark_p(
  type = "line",
  serie_index = 2,
  data = list(
    list(xAxis = bb$day[10], yAxis = bb$SMI[10]),
    list(xAxis = bb$day[37], yAxis = bb$SMI[37])
  ),
  lineStyle = list(type = "solid", color = "yellow")
)e_mark_p(
  type = "area",
  serie_index = 1,
  data = list(
    list(xAxis = bb$day[95]),
    list(xAxis = bb$day[105])
  ),
  itemStyle = list(color = "lightblue"),
  label = list(formatter = "X-area", position = "middle")
)e_merge()
)

react <- eventReactive(input$tln, {
  tmp <- bb
  if (input$tln) tmp <- tmp |> group_by(day < 75)

tmp |> 
  e_charts(
    day,
    backgroundColor = "#181818",
    legend = list(textStyle = list(color = "#aaa")),
    timeline = input$tln
  ) |> 
  e_y_axis(scale = TRUE, axisLabel = list(color = "#aaa")) |> 
  e_line(CAC, symbol = "none", color = "#ff33b8") |> 
  e_line(SMI, symbol = "none", color = "green")
})
if (interactive()) {
  shinyApp(ui, server)
}
```
Description
Mark points and lines.

Usage

\begin{verbatim}
e_mark_point(
e, serie = NULL, data = NULL, 
   ...,
   title = NULL,
   title_position = NULL
)
e_mark_line(
e, serie = NULL, data = NULL, 
   ...,
   title = NULL,
   title_position = NULL
)
e_mark_area(
e, serie = NULL, data = NULL, 
   ...,
   title = NULL,
   title_position = NULL
)
\end{verbatim}

Arguments
\begin{itemize}
  \item \textbf{e} \hspace{1cm} An echarts4r object as returned by \texttt{e_charts} or a proxy as returned by \texttt{echarts4rProxy}.
  \item \textbf{serie} \hspace{1cm} Serie or vector of series to mark on, defaults to all series.
  \item \textbf{data} \hspace{1cm} Placement of point, line or area.
  \item \textbf{...} \hspace{1cm} Any other option to pass, check See Also section.
  \item \textbf{title} \hspace{1cm} A convenience argument to easily set label, see details.
  \item \textbf{title_position} \hspace{1cm} Position of title.
\end{itemize}

Details
To set a label you need to either use the \texttt{title} argument or pass a list specifying the label formatter. \texttt{label = list(formatter = "label")}. The former is more convenient but more limited, e.g.: you cannot specify the placement of the label. When the \texttt{e_mark} series function is used with \texttt{e_timeline} at the same time, if the number of marks provided does not match the series, the mark information will follow the setting of the previous frame.
See Also

Additional point arguments, Additional line arguments

Examples

```r
max <- list(
  name = "Max",
  type = "max"
)

min <- list(
  name = "Min",
  type = "min"
)

avg <- list(
  type = "average",
  name = "AVG"
)

mtcars |>
  e_charts(mpg) |>
  e_line(wt) |>
  e_line(drat) |>
  e_line(cyl) |>
  e_mark_point("wt", data = max) |>
  e_mark_point(c("cyl", "drat"), data = min) |>
  e_mark_line(data = avg) |> # applies to all
  e_mark_area(
    serie = "wt",
    data = list(
      list(xAxis = "min", yAxis = "min"),
      list(xAxis = "max", yAxis = "max")
    )
  )
)

# Serie options, since the mark of "virginica" is not set, the mark setting
# of the previous frame is used
iris |>  
  group_by(Species) |>  
  e_charts(Sepal.Length, timeline = TRUE) |>  
  e_line(Sepal.Width) |>  
  e_timeline_serie(
    title = list(
      list(text = "setosa"),
      list(text = "versicolor"),
      list(text = "virginica")
    )
  ) |>
  e_mark_area(
    serie = "setosa",
    data = list(
      list(xAxis = "min", yAxis = "min"),
      list(xAxis = "max", yAxis = "max"
    )
  )
```
```r
list(xAxis = 4, yAxis = 2),
list(xAxis = 6, yAxis = 4.5)
),
itemStyle = list(color = "lightgreen")
)> e_mark_area(
   serie = "versicolor",
    data = list(
        list(xAxis = 4.5),
        list(xAxis = 7)
    ),
    itemStyle = list(color = "lightblue")
)
```

---

**e_merge**

*Merge options in chart, used in e_mark*

**Description**

Merge options in chart, used in e_mark

**Usage**

```r
e_merge(proxy)
```

**Arguments**

- `proxy` 
  An echarts4r proxy as returned by `echarts4rProxy`.

---

**e_modularity**

*Modularity*

**Description**

Graph modularity extension will do community detection and partition a graph’s vertices in several subsets. Each subset will be assigned a different color.

**Usage**

```r
e_modularity(e, modularity = TRUE)
```

**Arguments**

- `e` 
  An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `modularity` 
  Either set to TRUE, or a list.
Modularity

- resolution Resolution
- sort Whether to sort to communities

Note

Does not work in RStudio viewer, open in browser.

See Also

Official documentation

Examples

```r
nodes <- data.frame(
  name = paste0(LETTERS, 1:100),
  value = rnorm(100, 10, 2),
  stringsAsFactors = FALSE
)
edges <- data.frame(
  source = sample(nodes$name, 200, replace = TRUE),
  target = sample(nodes$name, 200, replace = TRUE),
  stringsAsFactors = FALSE
)
e_charts() |>
e_graph() |>
e_graph_nodes(nodes, name, value) |>
e_graph_edges(edges, source, target) |>
e_modularity(
  list(
    resolution = 5,
    sort = TRUE
  )
)
```

---

e_morph  Morphing

Description

__This is experimental__

Usage

```
e_morph(e, ..., callback, default = 1L)
```
e_parallel

Arguments

- `e, ...` - Graphs (from `e_graph`).
- `callback` - JavaScript callback function as a character string (vector of length 1). This function has access to the `chart` object, as well as `opts` an array containing the options of the charts passed to `e` and `...`.
- `default` - Default chart to show.

Details

Morph between graphs.

Examples

```r
mtcars2 <- mtcars |> 
  head() |> 
  tibble::rownames_to_column("model")

e1 <- mtcars2 |> 
  e_charts(model) |> 
  e_bar( 
    carb, 
    universalTransition = TRUE, 
    animationDurationUpdate = 1000L
  )

e2 <- mtcars2 |> 
  e_charts(model) |> 
  e_pie( 
    carb, 
    universalTransition = TRUE, 
    animationDurationUpdate = 1000L
  )

cb <- "() => { 
  let x = 0; 
  setInterval(() => { 
    x++ 
    chart.setOption(opts[x % 2], true); 
  }, 3000); 
}"

e_morph(e1, e2, callback = cb)
```

---

e_parallel

Parallel
Description

Draw parallel coordinates.

Usage

e_parallel(e, ..., name = NULL, rm_x = TRUE, rm_y = TRUE, opts = list())
e_parallel_(e, ..., name = NULL, rm_x = TRUE, rm_y = TRUE, opts = list())

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.

... Columns to select from the data passed to e_charts.

name name of the serie.

rm_x, rm_y Whether to remove x and y axis, defaults to TRUE.

opts A list of additional options to pass to the serie.

See Also

Additional arguments

Examples

df <- data.frame(
  price = rnorm(5, 10),
  amount = rnorm(5, 15),
  letter = LETTERS[1:5]
)

df |> 
e_charts() |> 
e_parallel(price, amount, letter, opts = list(smooth = TRUE))

Description

Pictorial bar chart is a type of bar chart that customzied glyph (like images, SVG PathData) can be used instead of rectangular bar.
Usage

e_pictorial(
  e,
  serie,
  symbol,
  bind,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  ...
)

e_pictorial_(
  e,
  serie,
  symbol,
  bind = NULL,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  ...
)

Arguments

e        An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie    Column name of serie to plot.
symbol   Symbol to plot.
bind     Binding between datasets, namely for use of e_brush.
name     name of the serie.
legend   Whether to add serie to legend.
x_index, y_index
          Indexes of x and y axis.
...      Any other option to pass, check See Also section.

Symbols

  • Built-in circle, rect, roundRect, triangle, diamond, pin, arrow.
  • SVG Path
  • Images Path to image, don’t forget to precede it with image://, see examples.

See Also

  Additional arguments
Examples

# built-in symbols
y <- rnorm(10, 10, 2)
df <- data.frame(
  x = 1:10,
  y = y,
  z = y - rnorm(10, 5, 1)
)

df |> 
e_charts(x) |> 
e_bar(z, barWidth = 10) |> 
e_pictorial(
  y,
  symbol = "rect",
  symbolRepeat = TRUE,
  z = -1,
  symbolSize = c(10, 4)
) |> 
e_theme("westeros")

# svg path
path <- "path://M0,10 L10,10 C5.5,10 5.5,5 5,0 C4.5,5 4.5,10 0,10 z"

style <- list(
  normal = list(opacity = 0.5),
  # normal
  emphasis = list(opacity = 1) # on hover
)

df |> 
e_charts(x) |> 
e_pictorial(
  y,
  symbol = path,
  barCategoryGap = "-130%",
  itemStyle = style
)

# image
# might not work in RStudio viewer
# open in browser
qomo <- paste0("https://ecomfe.github.io/echarts-examples/public/",
  "data/asset/img/hill-Qomolangma.png")
kili <- paste0(
  "https://ecomfe.github.io/echarts-examples/public/",
  "data/asset/img/hill-Kilimanjaro.png")
data <- data.frame(
  x = c("Qomolangma", "Kilimanjaro"),
  value = c(8844, 5895),
  symbol = c(
    paste0("image://", qomo),
    paste0("image://", kili)
  )
)

data |>
e_charts(x) |>
e_pictorial(value, symbol) |>
e_legend(FALSE)

# timeline
df <- data.frame(
  x = rep(1:5, 2),
  y = runif(10, 1, 10),
  year = c(
    rep(2017, 5),
    rep(2018, 5)
  )
)

df |>
  group_by(year) |>
  e_charts(x, timeline = TRUE) |>
  e_pictorial(
    y,
    symbol = "rect",
    symbolRepeat = TRUE,
    z = -1,
    symbolSize = c(10, 4)
  )

---

**e_pie**

**Pie**

**Description**

Draw pie and donut charts.

**Usage**

```r
e_pie(e, serie, name = NULL, legend = TRUE, rm_x = TRUE, rm_y = TRUE, ...)
e_pie_(e, serie, name = NULL, legend = TRUE, rm_x = TRUE, rm_y = TRUE, ...)
```
Arguments

\[\text{e} \quad \text{An \texttt{echarts4r} object as returned by \texttt{e\_charts} or a proxy as returned by \texttt{echarts4rProxy}.}\]

\[\text{serie} \quad \text{Column name of serie to plot.}\]

\[\text{name} \quad \text{name of the serie.}\]

\[\text{legend} \quad \text{Whether to add serie to legend.}\]

\[\text{rm\_x, rm\_y} \quad \text{Whether to remove x and y axis, defaults to \texttt{TRUE}.}\]

\[... \quad \text{Any other option to pass, check See Also section.}\]

See Also

Additional arguments

Examples

\[\text{mtcars |> head()} |> \]
\[\text{tibble::rownames\_to\_column("model") |> e\_charts(model) |> e\_pie(carb)}\]

\[# timeline\]
\[\text{df <- data.frame(}\]
\[\text{labels = rep(LETTERS[1:3], 2),}\]
\[\text{values = runif(6, 1, 5)}\]
\[\text{)}\]
\[\text{df |> group\_by(grp) |> e\_charts(labels, timeline = \texttt{TRUE}) |> e\_pie(values)}\]

---

\[\text{e\_polar} \quad \text{Polar}\]

Description

Customise polar coordinates.

Usage

\[\text{e\_polar(e, show = \texttt{TRUE}, \ldots)}\]

Arguments

\[\text{e} \quad \text{An \texttt{echarts4r} object as returned by \texttt{e\_charts} or a proxy as returned by \texttt{echarts4rProxy}.}\]

\[\text{show} \quad \text{Whether to display the axis.}\]

\[... \quad \text{Any other option to pass, check See Also section.}\]
See Also

Additional arguments

Examples

```r
df <- data.frame(x = 1:10, y = seq(1, 20, by = 2))

df |> e_charts(x) |> e_polar() |> e_angle_axis() |> e_radius_axis() |> e_line(y, coord.system = "polar", smooth = TRUE)
```

---

**e_radar**

**Radar**

Description

Add a radar chart

Usage

```r
e_radar(
  e,
  serie,
  max = 100,
  name = NULL,
  legend = TRUE,
  rm_x = TRUE,
  rm_y = TRUE,
  ...,
  radar = list()
)
```

```r
e_radar_(
  e,
  serie,
  max = 100,
  name = NULL,
  legend = TRUE,
  rm_x = TRUE,
  rm_y = TRUE,
  ...,
  radar = list()
)
```
**Arguments**

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **serie**: Column name of serie to plot.
- **max**: Maximum value.
- **name**: Name of the serie.
- **legend**: Whether to add serie to legend.
- **rm_x, rm_y**: Whether to remove x and y axis, defaults to `TRUE`.
- **...**: Any other option to pass, check See Also section.
- **radar**: A list of options to pass to the `radar` rather than the serie, see official documentation alternatively, use the `e_radar_opts`.

**Examples**

```r
df <- data.frame(
  x = LETTERS[1:5],
  y = runif(5, 1, 5),
  z = runif(5, 3, 7)
)

df |> 
  e_charts(x) |> 
  e_radar(y, max = 7) |> 
  e_radar(z) |> 
  e_tooltip(trigger = "item")
```

---

<table>
<thead>
<tr>
<th><code>e_radar_opts</code></th>
<th><strong>Radar axis</strong></th>
</tr>
</thead>
</table>

**Description**

Radar axis setup and options.

**Usage**

`e_radar_opts(e, index = 0, ...)`

**Arguments**

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **index**: Index of axis to customise.
- **...**: Any other option to pass, check See Also section.
Examples

df <- data.frame(
  x = LETTERS[1:5],
  y = runif(5, 1, 5),
  z = runif(5, 3, 7)
)

df |> 
  e_charts(x) |> 
  e_radar(y, max = 7) |> 
  e_radar(z) |> 
  e_radar_opts(center = c("25\%", "25\%")) |> 
  e_tooltip(trigger = "item")

---

Description

Remove a serie by name or precising its index.

Usage

e_remove_serie_p(proxy, serie_name = NULL, serie_index = NULL)
e_remove_serie(proxy, serie_name = NULL, serie_index = NULL)

Arguments

proxy An echarts4r proxy as returned by echarts4rProxy.
serie_name Name of serie to remove.
serie_index Index of serie to append to (starts from 0).

Examples

library(shiny)

ui <- fluidPage(
  actionButton("rm", "Remove z serie"),
  echarts4rOutput("plot")
)

server <- function(input, output, session) {
  data <- data.frame(
    x = rnorm(10, 5, 3),
    y = rnorm(10, 50, 12),
    z = rnorm(10, 50, 5)
  )
}
```r
output$plot <- renderEcharts4r(
  data |>
  e_charts(x) |>
  e_scatter(y) |>
  e_scatter(z)
)

observeEvent(input$rm, {
  echarts4rProxy("plot") |>
  e_remove_serie_p(serie_name = "z")
})

## Not run:
shinyApp(ui, server)

## End(Not run)
```

---

**e_resize**

<table>
<thead>
<tr>
<th>Resize</th>
</tr>
</thead>
</table>

**Description**

Force resize the chart.

**Usage**

```r
e_resize(proxy)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy</td>
<td>An echarts4r proxy as returned by <code>echarts4rProxy</code>.</td>
</tr>
</tbody>
</table>

---

**e_restore**

| Restore Toolbox |

**Description**

Restore Toolbox.

**Usage**

```r
e_restore(e, btn = NULL)
```
Arguments

e  An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
btn  A e_button id.

Examples

cars |>
  e_charts(speed) |>
  e_scatter(dist) |>
  e_datazoom() |>
  e_restore("btn") |>
  e_button("btn", "Reset")

Description

Build a theme river.

Usage

e_river(e, serie, name = NULL, legend = TRUE, rm_x = TRUE, rm_y = TRUE, ...)
e_river_(e, serie, name = NULL, legend = TRUE, rm_x = TRUE, rm_y = TRUE, ...)

Arguments

e  An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie  Column name of serie to plot.
name  name of the serie.
legend  Whether to add serie to legend.
rm_x, rm_y  Whether to remove x and y axis, defaults to TRUE.
...  Any other option to pass, check See Also section.

See Also

Additional arguments
Examples

dates <- seq.Date(Sys.Date() - 30, Sys.Date(), by = "day")
grps <- lapply(LETTERS[1:3], rep, 31) |> unlist()

df <- data.frame(
  dates = rep(dates, 3),
  groups = grps,
  values = runif(length(grps), 1, 50)
)

df |> group_by(groups) |> e_charts(dates) |> e_river(values) |> e_tooltip(trigger = "axis")

---

e_sankey  Sankey

Description

Draw a sankey diagram.

Usage

e_sankey(
  e,
  source,
  target,
  value,
  layout = "none",
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_sankey_
  e,
  source,
  target,
  value,
  layout = "none",
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
e_scatter

Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **source, target**: Source and target columns.
- **value**: Value change from source to target.
- **layout**: Layout of sankey.
- **rm_x, rm_y**: Whether to remove the x and y axis, defaults to TRUE.
- **...**: Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

```r
sankey <- data.frame(
  source = c("a", "b", "c", "d", "c"),
  target = c("b", "c", "d", "e", "e"),
  value = ceiling(rnorm(5, 10, 1)),
  stringsAsFactors = FALSE
)

sankey |> e_charts() |> e_sankey(source, target, value)
```

---

### e_scatter

**Scatter**

Description

Add scatter serie.

Usage

```r
e_scatter(
  e,
  serie,
  size,
  bind,
  symbol = NULL,
  symbol_size = 1,
  scale = e_scale,
  scale_js = "function(data){ return data[3];}",
  name = NULL,
  coord_system = "cartesian2d",
  jitter_factor = 0,
)```
jitter_amount = NULL,
legend = TRUE,
y_index = 0,
x_index = 0,
rm_x = TRUE,
rm_y = TRUE,
...
)

e_effect_scatter(
  e,
  serie,
  size,
  bind,
  symbol = NULL,
  symbol_size = 1,
  scale = e_scale,
  scale_js = "function(data){ return data[3];}",
  name = NULL,
  coord_system = "cartesian2d",
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_scale(x)

e_scatter_(
  e,
  serie,
  size = NULL,
  bind = NULL,
  symbol = NULL,
  symbol_size = 1,
  scale = e_scale,
  scale_js = "function(data){ return data[3];}",
  name = NULL,
  coord_system = "cartesian2d",
  jitter_factor = 0,
  jitter_amount = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  rm_x = TRUE,
  rm_y = TRUE,
e_effect_scatter_(
    e,
    serie,
    size = NULL,
    bind = NULL,
    symbol = NULL,
    symbol_size = 1,
    scale = e_scale,
    scale_js = "function(data){ return data[3];}",
    name = NULL,
    coord_system = "cartesian2d",
    legend = TRUE,
    y_index = 0,
    x_index = 0,
    rm_x = TRUE,
    rm_y = TRUE,
    ...
)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie Column name of serie to plot.
size Column name containing size of points.
bind Binding between datasets, namely for use of e_brush.
symbol The symbol to use, default to NULL, can also be circle, rect, roundRect, triangle, diamond, pin, arrow, or none.
symbol_size Size of points, either an integer or a vector of length 2, if size is not NULL or missing it is applied as a multiplier to scale.
scale A function that takes a vector of numeric and returns a vector of numeric of the same length. You can disable the scaling by setting it to NULL.
scale_js the JavaScript scaling function.
name name of the serie.
coord_system Coordinate system to plot against, see examples.
jitter_factor, jitter_amount Jitter points, passed to jitter.
legend Whether to add serie to legend.
x_index, y_index Indexes of x and y axis.
rm_x, rm_y Whether to remove x and y axis, only applies if coord_system is not set to cartesian2d.
... Any other option to pass, check See Also section.
x A vector of integers or numeric.
Scaling function

defaults to `e_scale` which is a basic function that rescales size between 1 and 20 for that makes for decent sized points on the chart.

See Also

Additional arguments scatter, Additional arguments for effect scatter

Examples

```r
# scaling
e_scale(c(1, 1000))

mtcars |>  
  e_charts(mpg) |>  
  e_scatter(wt, qsec)

# custom function
my_scale <- function(x) scales::rescale(x, to = c(2, 50))
echart <- mtcars |>  
  e_charts(mpg) |>  
  e_scatter(wt, qsec, scale = my_scale)

echart

# rescale color too
echart |>  
  e_visual_map(wt, scale = my_scale)

# or
echart |>  
  e_visual_map(min = 2, max = 50)

# disable scaling
mtcars |>  
  e_charts(qsec) |>  
  e_scatter(wt, mpg, scale = NULL)

# jitter point
mtcars |>  
  e_charts(cyl) |>  
  e_scatter(wt, symbol_size = 5) |>  
  e_scatter(wt, jitter_factor = 2, legend = FALSE)

# examples
USArrests |>  
  e_charts(Assault) |>  
  e_scatter(Murder, Rape) |>  
  e_effect_scatter(Rape, Murder, y_index = 1) |>  
  e_grid(index = c(0, 1)) |>  
  e_tooltip()
```
iris |>
  e_charts("Sepal.Length") |>
  e_scatter_(
    "Sepal.Width",
    symbol_size = c(8, 2),
    symbol = "rect"
  ) |>
  e_x_axis(min = 4)

quakes |>
  e_charts(long) |>
  e_geo(
    roam = TRUE,
    boundingCoords = list(
      c(185, -10),
      c(165, -40)
    )
  ) |>
  e_scatter(lat, mag, coord_system = "geo") |>
  e_visual_map(min = 4, max = 6.5)

# timeline
iris |>
  group_by(Species) |>
  e_charts(Petal.Width, timeline = TRUE) |>
  e_scatter(Sepal.Width, Sepal.Length) |>
  e_tooltip(trigger = "axis")

---

**e_scatter_3d Scatter 3D**

**Description**

Add 3D scatter.

**Usage**

```r

```
e_scatter_3d

legend = FALSE,
...
)

e_scatter_3d_(
e, y, z, color = NULL, size = NULL, bind = NULL, coord_system = "cartesian3D", name = NULL, rm_x = TRUE, rm_y = TRUE, legend = FALSE,
...
)

Arguments

e                      An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
y, z                   Coordinates.
color, size            Color and Size of bubbles.
bind                   Binding.
coord_system           Coordinate system to use, one of geo3D, globe, or cartesian3D.
name                   name of the serie.
rm_x, rm_y             Whether to remove x and y axis, defaults to TRUE.
legend                 Whether to add serie to legend.
...                    Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

v <- LETTERS[1:10]
matrix <- data.frame(
    x = sample(v, 300, replace = TRUE),
    y = sample(v, 300, replace = TRUE),
    z = rnorm(300, 10, 1),
    color = rnorm(300, 10, 1),
    size = rnorm(300, 10, 1),
    stringsAsFactors = FALSE
) |> dplyr::group_by(x, y) |> dplyr::summarise(
z = sum(z),
  color = sum(color),
  size = sum(size)
) |> 
dplyr::ungroup()

matrix |>
  e_charts(x) |>
  e_scatter_3d(y, z, size, color) |> 
e_visual_map(
    min = 1,
    max = 100,
    inRange = list(symbolSize = c(1, 30)),
    # scale size
    dimension = 3 # third dimension 0 = x, y = 1, z = 2, size = 3
  ) |> 
e_visual_map(
    min = 1,
    max = 100,
    inRange = list(color = c("#bf444c", "#d88273", "#f6efa6")),
    # scale colors
    dimension = 4,
    # third dimension 0 = x, y = 1, z = 2, size = 3, color = 4
    bottom = 300 # padding to avoid visual maps overlap
  )

airports <- read.csv(
paste0(
  "https://raw.githubusercontent.com/plotly/datasets/",
  "master/2011_february_us_airport_traffic.csv"
)
)

airports |>
  e_charts(long) |> 
e_globe(
    globeOuterRadius = 100
  ) |> 
e_scatter_3d(lat, cnt, coord_system = "globe", blendMode = "lighter") |> 
e_visual_map(inRange = list(symbolSize = c(1, 10)))

# timeline
airports |>
  group_by(state) |>
  e_charts(long, timeline = TRUE) |> 
e_globe(
    globeOuterRadius = 100
  ) |> 
e_scatter_3d(lat, cnt, coord_system = "globe", blendMode = "lighter") |> 
e_visual_map(inRange = list(symbolSize = c(1, 10)))
e_scatter_gl

Description

Draw scatter GL.

Usage

e_scatter_gl(
  e,
  y,
  z,
  name = NULL,
  coord_system = "geo",
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_scatter_gl_(
  e,
  y,
  z,
  name = NULL,
  coord_system = "geo",
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

Arguments

e          An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
y, z       Column names containing y and z data.
name       name of the serie.
coord_system Coordinate system to plot against.
rm_x, rm_y Whether to remove x and y axis, defaults to TRUE.
...        Any other option to pass, check See Also section.

See Also

Additional arguments
Examples

```r
quakes |> 
  e_charts(long) |> 
  e_geo( 
    roam = TRUE, 
    boundingCoords = list( 
      c(185, -10), 
      c(165, -40) 
    )
  ) |>
  e_scatter_gl(lat, depth)

# timeline
quakes$year <- rep(c("2017", "2018"), 500)

quakes |> 
  group_by(year) |> 
  e_charts(long, timeline = TRUE) |> 
  e_geo( 
    roam = TRUE, 
    boundingCoords = list( 
      c(185, -10), 
      c(165, -40) 
    )
  ) |>
  e_scatter_gl(lat, depth)
```

---

**e_showtip_p**  
*Tooltip Proxy*

**Description**

Proxies to show or hide tooltip.

**Usage**

```r
e_showtip_p(proxy, ...)
e_hidetip_p(proxy)
```

**Arguments**

- `proxy`  
  An echarts4r proxy as returned by `echarts4rProxy`.
- `...`  
  Any other option, see `showTip`.  

Examples

```r
## Not run:
library(shiny)

ui <- fluidPage(
  fluidRow(
    actionButton("show", "Show tooltip"),
    actionButton("hide", "Hide tooltip")
  ),
  fluidRow(
    echarts4rOutput("plot"),
    h3("clicked Data"),
    verbatimTextOutput("clickedData"),
    h3("clicked Serie"),
    verbatimTextOutput("clickedSerie"),
    h3("clicked Row"),
    verbatimTextOutput("clickedRow")
  )
)

server <- function(input, output, session) {
  output$plot <- renderEcharts4r({
    mtcars |>
      e_charts(mpg) |>
      e_line(disp, bind = carb, name = "displacement") |>
      e_line(hp) |>
      e_x_axis(min = 10) |>
      e_tooltip(show = FALSE) |>
      e_theme("westeros")
  })

  observeEvent(input$show, {
    echarts4rProxy("plot") |>
      e_showtip_p(
        name = "displacement",
        position = list(5, 5)
      )
  })

  observeEvent(input$hide, {
    echarts4rProxy("plot") |>
      e_hidetip_p()
  })

  output$clickedData <- renderPrint({
    input$plot_clicked_data
  })

  output$clickedSerie <- renderPrint({
    input$plot_clicked_serie
  })
}
```

output$clickedRow <- renderPrint({
  input$plot_clicked_row
})

if (interactive()) {
  shinyApp(ui, server)
}

## End(Not run)

e_show_loading

Description

Show or hide loading.

Usage

e_show_loading(e, 
  hide_overlay = TRUE, 
  text = "loading", 
  color = "#c23531", 
  text_color = "#000", 
  mask_color = "rgba(255, 255, 255, 0.8)", 
  zlevel = 0
)
e_hide_loading(e)

Arguments

e  An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
hide_overlay  Hides the white overaly that appears in shiny when a plot is recalculating.
text  Text to display.
color  Color of spinner.
text_color  Color of text.
mask_color  Color of mask.
zlevel  Z level.

Details

This only applies to Shiny.
Examples

## Not run:

```r
# no redraw
# no loading
library(shiny)
ui <- fluidPage(
    fluidRow(
        column(12, actionButton("update", "Update"))
    ),
    fluidRow(
        column(12, echarts4rOutput("plot"))
    )
)

server <- function(input, output) {
  data <- eventReactive(input$update, {
    data.frame(
        x = 1:10,
        y = rnorm(10)
    )
  })

  output$plot <- renderEcharts4r({
    data() |>
    e_charts(x) |>
    e_bar(y)
  })
}

if (interactive()) {
  shinyApp(ui, server)
}

# add loading
server <- function(input, output) {
  data <- eventReactive(input$update, {
    Sys.sleep(1) # sleep one second to show loading
    data.frame(
        x = 1:10,
        y = rnorm(10)
    )
  })

  output$plot <- renderEcharts4r({
    data() |>
    e_charts(x) |>
    e_bar(y) |>
    e_show_loading()
  })
}
```
if (interactive()) {
  shinyApp(ui, server)
}

## End(Not run)

---

e_single_axis | Single Axis

### Description

Setup single axis.

### Usage

e_single_axis(e, index = 0, ...)

### Arguments

e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.

index | Index of axis to customise.

... | Any other option to pass, check See Also section.

### Examples

def <- data.frame(
  axis = LETTERS[1:10],
  value = runif(10, 3, 20),
  size = runif(10, 3, 20)
)

def |> 
  e_charts(axis) |> 
  e_single_axis() |> # add the single axis 
  e_scatter(
    value, 
    size, 
    coord_system = "singleAxis"
)
Description

Add step serie.

Usage

e_step(
  e,
  serie,
  bind,
  step = c("start", "middle", "end"),
  fill = FALSE,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)

e_step_(
  e,
  serie,
  bind = NULL,
  step = c("start", "middle", "end"),
  fill = FALSE,
  name = NULL,
  legend = TRUE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  ...
)

Arguments

e          An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie      Column name of serie to plot.
bind       Binding between datasets, namely for use of e_brush.
step       Step type, one of start, middle or end.
fill       Set to fill as area.
name       name of the serie.
legend
Whether to add serie to legend.

x_index, y_index
Indexes of x and y axis.

coord_system
Coordinate system to plot against.

... Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

USArrests |>
tibble::rownames_to_column("State") |>
e_charts(State) |>
e_step(Murder, name = "Start", step = "start", fill = TRUE) |>
e_step(Rape, name = "Middle", step = "middle") |>
e_step(Assault, name = "End", step = "end") |>
e_tooltip(trigger = "axis")

# timeline
iris |>
  group_by(Species) |>
  e_charts(Sepal.Length, timeline = TRUE) |>
  e_step(Sepal.Width) |>
  e_tooltip(trigger = "axis")

---

e_sunburst

Sunburst

Description

Build a sunburst.

Usage

e_sunburst(
  e,
  styles = NULL,
  names = NULL,
  levels = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_sunburst_(
  e,
e\textunderscore\text{sunburst}

\begin{verbatim}
styles = NULL,
names = NULL,
levels = NULL,
rm\_x = TRUE,
rm\_y = TRUE,
...
)

Arguments
\begin{itemize}
\item \textbf{e} An \texttt{echarts4r} object as returned by \texttt{e\_charts} or a proxy as returned by \texttt{echarts4rProxy}.
\item \textbf{styles} Vector of style lists, defaults to \texttt{NULL}.
\item \textbf{names} Names of items to style, expects a \texttt{list}, defaults to \texttt{NULL}.
\item \textbf{levels} Hierarchical levels to style, expects a \texttt{list}, defaults to \texttt{NULL}.
\item \textbf{rm\_x, rm\_y} Whether to remove x and y axis, defaults to \texttt{TRUE}.
\item \ldots Any other option to pass, check See Also section.
\end{itemize}

Details
Charts \texttt{e\_sunburst}, \texttt{e\_treemap} and \texttt{e\_tree} require hierarchical input data. Such structure could be represented thru \texttt{json} lists or nested \texttt{tibbles (data.frame)}. Input data may contain styles, see itemStyle in examples \texttt{jsonl} and \texttt{df} below. The number of lists in the \texttt{styles} parameter should match the number of elements in \texttt{names} and/or \texttt{levels}. If both \texttt{names} and \texttt{levels} are present, name styles will take precedence over level styles. Multiple names may have the same style, see \texttt{c('land','river')} below. Multiple levels may have the same style, see \texttt{c(3,4)} below. \texttt{styles} lists contain items such as \texttt{color}, or \texttt{borderColor} as specified in the official documentation.

See Also
\texttt{Additional arguments}

Examples
\begin{verbatim}
# json list hierarchical data representation
jsonl <- jsonlite::fromJSON('[
{"name": "earth", "value": 30,
 "children": [
{"name": "land", "value":10,
 "children": [
{"name": "forest", "value": 3},
{"name": "river", "value": 7}
 ]},
{"name": "ocean", "value":20,
 "children": [
{"name": "fish", "value": 10,
 "children": [
{"name": "shark", "value":2},
{"name": "tuna", "value":6}

\end{verbatim}
```r
library(dplyr)

df <- tibble(
  name = c("earth", "mars", "venus"),
  value = c(30, 40, 30),
  # 1st level
  itemStyle = tibble(color = c(NA, "red", "blue")),
  # embedded styles, optional
  children = list(
    tibble(
      name = c("land", "ocean"),
      value = c(10, 20),
      # 2nd level
      children = list(
        tibble(name = c("forest", "river"), value = c(3, 7)),
        # 3rd level
        tibble(
          name = c("fish", "kelp"),
          value = c(10, 5),
          children = list(
            tibble(name = c("shark", "tuna"), value = c(2, 6)),
            # 4th level
            NULL # kelp
          )
        )
      )
    )
  ),
  tibble(name = c("crater", "valley"), value = c(20, 20)),
  NULL # venus
)

df |> e_charts() |> e_sunburst() |
```

---

RAW TEXT:

```json
{
  "name": "earth",
  "value": 30,
  "children": [
    {
      "name": "land",
      "value": 10,
      "children": [
        {
          "name": "forest",
          "value": 3,
        },
        {
          "name": "river",
          "value": 7,
        }
      ]
    },
    {
      "name": "ocean",
      "value": 20,
      "children": [
        {
          "name": "fish",
          "value": 10,
          "children": [
            {
              "name": "shark",
              "value": 2,
            },
            {
              "name": "tuna",
              "value": 6,
            }
          ]
        },
        {
          "name": "kelp",
          "value": 5
        }
      ]
    }
  ]
}
```

---

The given R code demonstrates creating a hierarchical data structure and converting it to a json object. The code then visualizes this data using the `e_charts()` function and `e_sunburst()` function provided by a package named `echarts`. The hierarchical data is represented in a tibble, with nested levels for each category. The `itemStyle` column is used to assign colors to different levels of the hierarchy. The final visualizations will show the data in a sunburst chart format, with each level of the hierarchy represented as a segment of the chart.
e_theme("westeros")

# with styles
myStyles <- c(list(color = "green"), list(color = "magenta")) # custom styles defined
myNames <- list(c("land", "river"), "crater") # names to style
myLevels <- list(2, c(3, 4)) # hierarchical levels to style

df |>  
    e_charts() |>  
    e_sunburst(myStyles, myNames, myLevels)

---

**e_surface**

**Surface**

**Description**

Add a surface plot.

**Usage**

```r

```

**Arguments**

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `y, z`: Coordinates.
- `bind`: Binding.
- `name`: name of the serie.
- `rm_x, rm_y`: Whether to remove x and y axis, defaults to TRUE.
- `...`: Any other option to pass, check See Also section.

**Examples**

```r
data("volcano")

surface <- as.data.frame(as.table(volcano))
surface$Var1 <- as.numeric(surface$Var1)
surface$Var2 <- as.numeric(surface$Var2)

surface |>  
    e_charts(Var1) |>  
    e_surface(Var2, Freq) |>  
    e_visual_map(Freq)
```
**e_text_style**

*Text style*

**Description**

Define global font style.

**Usage**

```r
e_text_style(e, ...)
```

**Arguments**

- `e` An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `...` Any other option to pass, check See Also section.

**Note**

Do not use `e_arrange` in R markdown or Shiny.

**See Also**

- [official documentation](#)

**Examples**

```r
cars |> e_charts(dist) |> e_scatter(speed) |> e_labels() |> e_text_style(  
  color = "blue",  
  fontStyle = "italic"  
)
```

---

**e_theme**

*Themes*

**Description**

Add a custom theme or apply a pre-built one.
Usage

e_theme(e,  
  name = c("auritus", "azul", "bee-inspired", "blue", "caravan", "carp", "chalk", "cool",  
  "dark-blue", "dark-bold", "dark-digerati", "dark-fresh-cut", "dark-mushroom", "dark",  
  "eduardo", "essos", "forest", "fresh-cut", "fruit", "gray", "green", "halloween",  
  "helianthus", "infographic", "inspired", "jazz", "london", "macarons", "macarons2",  
  "mint", "purple-passion", "red-velvet", "red", "roma", "royal", "sakura", "shine",  
  "tech-blue", "vintage", "walden", "wef", "weforum", "westeros", "wonderland")  
)

e_theme_custom(e, theme, name = "custom")

e_theme_register(theme, name = "custom")

Arguments

e         An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
name      Name of theme.
theme     Theme, A json string or a see below.

Details

The function e_theme_register can be used to register the theme globally in R markdown or shiny (UI). This is useful because 1) the e_theme_custom registers the theme every time and is more computationally expensive.

Functions

- e_theme - Use a default theme by name.
- e_theme_custom - Use a custom theme.
- e_theme_register - Register a theme globally in shiny or R markdown.

See Also

create your own theme.

Examples

mtcars |>  
  e_charts(mpg) |>  
  e_line(disp) |>  
  e_area(hp) |>  
  e_x_axis(min = 10) -> p

p |> e_theme("chalk")  
p |> e_theme_custom(‘"color":["#ff715e","ffaf51"]’)
### e_title

**Title**

#### Description

Add title.

#### Usage

```r
e_title(e, text = NULL, subtext = NULL, link = NULL, sublink = NULL, ...)
```

#### Arguments

- `e` An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `text, subtext` Title and Subtitle.
- `link, sublink` Title and Subtitle link.
- `...` Any other option to pass, check See Also section.

#### See Also

Additional arguments

#### Examples

```r
quakes |>
  dplyr::mutate(mag = exp(mag) / 60) |>
  e_charts(stations) |>
  e_scatter(depth, mag) |>
  e_visual_map(min = 3, max = 7) |>
  e_title("Quakes", "Stations and Magnitude")
```

### e_toolbox_feature

**Toolbox**

#### Description

Add toolbox interface.

#### Usage

```r
e_toolbox_feature(e, feature, ...)
e_toolbox(e, ...)
```
Arguments

- **e**: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **feature**: Feature to add, defaults to all.
- **...**: Any other option to pass, check See Also section.

Details

Valid feature:

- `saveAsImage`
- `brush`
- `restore`
- `dataView`
- `dataZoom`
- `magicType`

See Also

Additional arguments

Examples

```r
USArrests |>  
  e_charts(UrbanPop) |>  
  e_line(Assault) |>  
  e_area(Murder, y_index = 1, x_index = 1) |>  
  e_datazoom(x_index = 0)

mtcars |>  
  tibble::rownames_to_column("model") |>  
  e_charts(model) |>  
  e_line(qsec) |>  
  e_toolbox() |>  
  e_toolbox_feature(  
    feature = "magicType",  
    type = list("line", "bar")  
  )
```

Description

Customise tooltip
Usage

e_tooltip(e, trigger = c("item", "axis"), formatter = NULL, ...)

e_tooltip_item_formatter(
  style = c("decimal", "percent", "currency"),
  digits = 0,
  locale = NULL,
  currency = "USD"
)

e_tooltip_choro_formatter(
  style = c("decimal", "percent", "currency"),
  digits = 0,
  locale = NULL,
  currency = "USD"
)

e_tooltip_pie_formatter(
  style = c("decimal", "percent", "currency"),
  digits = 0,
  locale = NULL,
  currency = "USD",
  ...
)

e_tooltip_pointer_formatter(
  style = c("decimal", "percent", "currency"),
  digits = 0,
  locale = NULL,
  currency = "USD"
)

Arguments

e       An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
trigger  What triggers the tooltip, one of item or axis.
formatter Item and Pointer formatter as returned by e_tooltip_item_formatter, e_tooltip_pointer_formatter, e_tooltip_pie_formatter.
...      Any other option to pass, check See Also section.
style   Formatter style, one of decimal, percent, or currency.
digits  Number of decimals.
locale  Locale, if NULL then it is inferred from Sys.getlocale.
currency Currency to to display.

Formatters

- e_tooltip_pie_formatter: special helper for e_pie.
• `e_tooltip_item_formatter`: general helper, this is passed to the `tooltip formatter`.
• `e_tooltip_pointer_formatter`: helper for pointer, this is passed to the `label parameter` under `axisPointer`.

See Also

Additional arguments

Examples

```r
# basic
USA Arrests |>
  e_charts(Assault) |>
  e_scatter(Murder) |>
  e_tooltip()

# formatter
cars |>
  dplyr::mutate(
    dist = dist / 120
  ) |>
  e_charts(speed) |>
  e_scatter(dist, symbol_size = 5) |>
  e_tooltip(
    formatter = e_tooltip_item_formatter("percent")
  )

# axis pointer
cars |>
  e_charts(speed) |>
  e_scatter(dist, symbol_size = 5) |>
  e_tooltip(
    formatter = e_tooltip_pointer_formatter("currency"),
    axisPointer = list(
      type = "cross"
    )
  )
```

**Description**

Build a tree.

**Usage**

```r
e_tree(e, rm_x = TRUE, rm_y = TRUE, ...)
e_tree_(e, rm_x = TRUE, rm_y = TRUE, ...)
```
Arguments

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `rm_x`, `rm_y`: Whether to remove x and y axis, defaults to TRUE.
- `...`: Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

```r
library(dplyr)
df <- tibble(
  name = "earth",
  # 1st level
  children = list(
    tibble(
      name = c("land", "ocean"),
      # 2nd level
      children = list(
        tibble(name = c("forest", "river")),
        # 3rd level
        tibble(
          name = c("fish", "kelp"),
          children = list(
            tibble(
              name = c("shark", "tuna")
            ),
            # 4th level
            NULL # kelp
          )
        )
      )
    )
  )
)

df |> e_charts() |> e_tree(initialTreeDepth = 3, label = list(offset = c(0, -11)))
```

Description

Build a treemap.
Usage

e_treemap(
  e,
  styles = NULL,
  names = NULL,
  levels = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

e_treemap_(
  e,
  styles = NULL,
  names = NULL,
  levels = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
styles Vector of style lists, defaults to NULL.
names Names of items to style, expects a list, defaults to NULL.
levels Hierarchical levels to style, expects a list, defaults to NULL.
rm_x, rm_y Whether to remove x and y axis, defaults to TRUE.
...
Any other option to pass, check See Also section.

See Also

Additional arguments

Examples

library(dplyr)
df <- tibble(
  name = c("earth", "mars", "venus"),
  value = c(30, 40, 30),
  # 1st level
  itemStyle = tibble(color = c(NA, "red", "blue")),
  # embedded styles, optional
  children = list(
    tibble(
      name = c("land", "ocean"),
      value = c(10, 20),
      # 2nd level
    )
  )
)
```r
children = list(
    tibble(name = c("forest", "river"), value = c(3, 7)),
    # 3rd level
tibble(
        name = c("fish", "kelp"),
        value = c(10, 5),
        children = list(
            tibble(name = c("shark", "tuna"), value = c(2, 6)),
            # 4th level
            NULL # kelp
        )
    ),
    tibble(name = c("crater", "valley"), value = c(20, 20)),
    NULL # venus
)
)

df |> 
    e_charts() |> 
    e_treemap()
```

---

**e_utc**

*Use UTC*

**Description**

Use UTC

**Usage**

```r
e_utc(e)
```

**Arguments**

- `e`  
  An echarts4r object as returned by `e_charts` or a proxy as returned by echarts4rProxy.

---

**e_visual_map**

*Visual Map*

**Description**

Visual Map
Usage

e_visual_map(
  e,
  serie,
  calculable = TRUE,
  type = c("continuous", "piecewise"),
  scale = NULL,
  ...
)

e_visual_map_(
  e,
  serie = NULL,
  calculable = TRUE,
  type = c("continuous", "piecewise"),
  scale = NULL,
  ...
)

Arguments

e                  An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
serie              Column name of serie to scale against.
calculable         Whether show handles, which can be dragged to adjust "selected range".
type               One of continuous or piecewise.
scale              A function that takes a vector of numeric and returns a vector of numeric of the same length.
...                  Any other option to pass, check See Also section.

Scaling function

defaults to e_scale which is a basic function that rescales size between 1 and 20 for that makes for decent sized points on the chart.

See Also

Additional arguments

Examples

# scaled data
mtcars |>  
  e_charts(mpg) |>  
  e_scatter(wt, qsec, scale = e_scale) |>  
  e_visual_map(qsec, scale = e_scale)

# dimension
# color according to y axis
mtcars |>
  e_charts(mpg) |>
  e_scatter(wt) |>
  e_visual_map(wt, dimension = 1)

# color according to x axis
mtcars |>
  e_charts(mpg) |>
  e_scatter(wt) |>
  e_visual_map(mpg, dimension = 0)

v <- LETTERS[1:10]
matrix <- data.frame(
  x = sample(v, 300, replace = TRUE),
  y = sample(v, 300, replace = TRUE),
  z = rnorm(300, 10, 1),
  color = rnorm(300, 10, 1),
  size = rnorm(300, 10, 1),
  stringsAsFactors = FALSE
) |>  
dplyr::group_by(x, y) |>  
dplyr::summarise(
  z = sum(z),
  color = sum(color),
  size = sum(size)
) |>  
dplyr::ungroup()

matrix |>
  e_charts(x) |>
  e_scatter_3d(y, z, color, size) |>
  e_visual_map(
    z,
    # scale to z
    inRange = list(symbolSize = c(1, 30)),
    # scale size
    dimension = 3 # third dimension 0 = x, y = 1, z = 2, size = 3
  ) |>  
e_visual_map(
    z,
    # scale to z
    inRange = list(color = c("#bf444c", "#d88273", "#f6efa6")),
    # scale colors
    dimension = 4,
    # third dimension 0 = x, y = 1, z = 2, size = 3, color = 4
    bottom = 300 # padding to avoid visual maps overlap
  )

---

e_visual_map_range  Select Visual Map
**e_zoom**

**Description**

Selects data range of visual mapping.

**Usage**

\[
\text{e\_visual\_map\_range}(e, \ldots, \text{btn} = \text{NULL})
\]

**Arguments**

- **e**: An `echarts4r` object as returned by `e\_charts` or a proxy as returned by `echarts4rProxy.`
- **\ldots**: Any options, see official documentation
- **btn**: A `e\_button` id.

**Examples**

```r
data("state")

as.data.frame(state.x77) |>
e\_charts(Population) |>
e\_scatter(Income, Frost) |>
e\_visual\_map(Frost, scale = e\_scale) |>
e\_legend(FALSE) |>
e\_visual\_map\_range(
  selected = list(60, 120)
)
```

---

**e_zoom**

**Zoom**

**Description**

Zoom on a region.

**Usage**

\[
\text{e\_zoom}(e, \ldots, \text{btn} = \text{NULL})
\]

**Arguments**

- **e**: An `echarts4r` object as returned by `e\_charts` or a proxy as returned by `echarts4rProxy.`
- **\ldots**: Any options, see official documentation
- **btn**: A `e\_button` id.
Examples

cars |> 
   e_charts(dist) |> 
   e_scatter(speed) |> 
   e_datazoom() |> 
   e_zoom(
      dataZoomIndex = 0, 
      start = 20, 
      end = 40, 
      btn = "BUTTON"
   ) |> 
   e_button("BUTTON", "Zoom in")

Description

Actions related to `e_graph`.

Usage

e_focus_adjacency(e, ..., btn = NULL)
e_unfocus_adjacency(e, ..., btn = NULL)

Arguments

e An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.

... Any options, see official documentation

btn A `e_button` id.

Examples

value <- rnorm(10, 10, 2)

nodes <- data.frame(
   name = sample(LETTERS, 10),
   value = value,
   size = value,
   grp = rep(c("grp1", "grp2"), 5),
   stringsAsFactors = FALSE
)

edges <- data.frame(
   source = sample(nodes$name, 20, replace = TRUE),
   target = sample(nodes$name, 20, replace = TRUE),
   stringsAsFactors = FALSE
)
highlight_action

)  

`e_charts() |>`  
`e_graph() |>`  
`e_graph_nodes(nodes, name, value, size, grp) |>`  
`e_graph_edges(edges, source, target) |>`  
`e_focus_adjacency(`  
  `seriesIndex = 0,`  
  `dataIndex = 4`  
`)`  

---

**highlight_action**  
**Highlight & Downplay**

**Description**

Highlight series

**Usage**

```r
e_highlight(e, series_index = NULL, series_name = NULL, btn = NULL)
```

```r
e_downplay(e, series_index = NULL, series_name = NULL, btn = NULL)
```

**Arguments**

- `e`: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `series_index, series_name`: Index or name of serie to highlight or list or vector of series.
- `btn`: A `e_button` id.

**Examples**

```r
iris |>  
  group_by(Species) |>  
  e_charts(Sepal.Length) |>  
  e_line(Sepal.Width) |>  
  e_line(Petal.Length) |>  
  e_highlight(series_name = "setosa") # highlight group
```
Description

Initialise a chart.

Usage

e_charts(
  data,
  x,
  width = NULL,
  height = NULL,
  elementId = NULL,
  dispose = TRUE,
  draw = TRUE,
  renderer = "canvas",
  timeline = FALSE,
  ...,
  reorder = TRUE
)

## Default S3 method:
e_charts(
  data,
  x,
  width = NULL,
  height = NULL,
  elementId = NULL,
  dispose = TRUE,
  draw = TRUE,
  renderer = "canvas",
  timeline = FALSE,
  ...,
  reorder = TRUE
)

## S3 method for class 'Node'
e_charts(
  data,
  x,
  width = NULL,
  height = NULL,
  elementId = NULL,
  dispose = TRUE,
  draw = TRUE,
renderer = "canvas",
timeline = FALSE,
..., 
reorder = TRUE
)

e_charts_(_
data,
x = NULL,
width = NULL,
height = NULL,
ele\ntId = NULL,
dispose = TRUE,
draw = TRUE,
renderer = "canvas",
timeline = FALSE,
..., 
reorder = TRUE
)

e_chart(
data,
x,
width = NULL,
height = NULL,
ele\ntId = NULL,
dispose = TRUE,
draw = TRUE,
renderer = "canvas",
timeline = FALSE,
..., 
reorder = TRUE
)

e_data(e, data, x)

Arguments

data A data.frame.
x Column name containing x axis.
width, height Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
ele\ntId Id of element.
dispose Set to TRUE to force redraw of chart, set to FALSE to update.
draw Whether to draw the chart, intended to be used with e_draw_p.
renderer Renderer, takes canvas (default) or svg.
timeline Set to TRUE to build a timeline, see timeline section.
...  Any other argument.
reorder  Set the FALSE to not reorder numeric x axis values.
e  An object of class echarts4r as returned by e_charts.

Timeline

The timeline feature currently supports the following chart types.

- e_bar
- e_line
- e_step
- e_area
- e_scatter
- e_effect_scatter
- e_candle
- e_heatmap
- e_pie
- e_line_3d
- e_lines_3d
- e_bar_3d
- e_lines
- e_scatter_3d
- e_scatter_gl
- e_histogram
- e_1m
- e_loess
- e_glm
- e_density
- e_pictorial
- e_boxplot
- e_map
- e_map_3d
- e_line_3d
- e_gauge
Examples

```r
mtcars |>
  e_charts(qsec) |>
  e_line(mpg)
points <- mtcars[1:3, ]
mtcars |>  
  e_charts_("qsec") |>
  e_line(mpg) |>
  e_data(points, qsec) |>  
  e_scatter(mpg, color = "red", symbol_size = 20)
```

---

**Legend**

Description

Legend

Usage

```r
e_legend_select(e, name, btn = NULL)
e_legend_unselect(e, name, btn = NULL)
e_legend_toggle_select(e, name, btn = NULL)
e_legend_scroll(e, scroll_index = NULL, legend_id = NULL, btn = NULL)
```

Arguments

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `name`: Legend name.
- `btn`: A `e_button` id.
- `scroll_index`: Control the scrolling of legend when `type = "scroll"` in `e_legend`.
- `legend_id`: Id of legend.

Examples

```r
e <- CO2 |
  group_by(Type) |>
  e_charts(conc) |>
  e_scatter(uptake)

  e |>
  e_legend_unselect("Quebec")

  e |>
  e_legend_unselect("Quebec", btn = "btn") |>  
  e_button("btn", "Quebec")
```
Description

Use mapbox.

Usage

e_mapbox(e, token, ...)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
token Your mapbox token from mapbox.
... Any option.

Note

Mapbox may not work properly in the RSudio console.

See Also

Official documentation, mapbox documentation

Examples

```r
## Not run:
url <- paste0(
  "https://echarts.apache.org/examples/",
  "data-gl/asset/data/population.json"
)
data <- jsonlite::fromJSON(url)
data <- as.data.frame(data)
names(data) <- c("lon", "lat", "value")
data |>
e_charts(lon) |>
e_mapbox(
  token = "YOUR_MAPBOX_TOKEN",
  style = "mapbox://styles/mapbox/dark-v9"
) |>
e_bar_3d(lat, value, coord_system = "mapbox") |>
e_visual_map()

## End(Not run)
```
Map Actions

Description

Map-related actions.

Usage

e_map_select(e, ..., btn = NULL)

e_map_unselect(e, ..., btn = NULL)

e_map_toggle_select(e, ..., btn = NULL)

Arguments

e An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy.
... Any options, see official documentation
btn A e_button id.

See Also

e_map_register

Examples

choropleth <- data.frame(countries = c("France", "Brazil", "China", "Russia", "Canada", "India", "United States", "Argentina", "Australia"),
values = round(runif(9, 10, 25)))

choropleth |>
e_charts(countries) |>
e_map(values) |>
e_visual_map(min = 10, max = 25) |>
e_map_toggle_select(name = "China", btn = "btn") |>
e_button("btn", "Select China")
Add nested data

**Description**

Utility function to add data where the original JavaScript library expects nested data.

**Usage**

```r
e_add(e, param, ..., .serie = NULL, .data = NULL)
e_add_nested(e, param, ..., .serie = NULL, .data = NULL)
e_add_unnested(e, param, value, .serie = NULL, .data = NULL)
```

**Arguments**

- `e`: An `echarts4r` object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `param`: The nested parameter to add data to.
- `...`: Any other option to pass, check See Also section.
- `.serie`: Serie's index to add the data to, if `NULL` then it is added to all.
- `.data`: A dataset to use, if none are specified than the original dataset passed to `e_charts` is used.
- `value`: The column to map to the parameter.

**Details**

For instance, `e_funnel` lets you pass values and labels (from your initial data.frame) which corresponds to name and value in the original library. However the latter also takes, `label`, `itemStyle`, and `emphasis` but being JSON arrays they translate to lists in R and dealing with nested data.frames is not ideal. `e_add` remedies to that. It allows adding those nested data points, see the examples below.

**Functions**

- `e_add_nested`: Adds nested data, e.g.: `e_add_nested("itemStyle", color, fontBold)` creates `itemStyle: color: 'red', fontBold: 'bold'`.
- `e_add_unnested`: Adds unnested data, e.g.: `e_add_unnested("symbolSize", size)` creates `symbolSize: 4`.

**Examples**

```r
# funnel can take nested itemStyle
# https://echarts.apache.org/en/option.html#series-funnel.data
funnel <- data.frame(
  stage = c("View", "Click", "Purchase"),
  value = c(80, 30, 20),
)
pie_action

```r

color = c("blue", "red", "green")

funnel |> e_charts() |> e_funnel(value, stage) |> e_add_nested("itemStyle", color)

# Heatmap can take nested label
# https://echarts.apache.org/en/option.html#series-heatmap.data
v <- LETTERS[1:10] matrix <- data.frame(x = sample(v, 300, replace = TRUE), y = sample(v, 300, replace = TRUE), z = rnorm(300, 10, 1), stringsAsFactors = FALSE) |> dplyr::group_by(x, y) |> dplyr::summarise(z = sum(z)) |> dplyr::ungroup() |> dplyr::mutate(show = TRUE, fontStyle = round(runif(dplyr::n(), 5, 12)))

matrix |> e_charts(x) |> e_heatmap(y, z) |> e_visual_map(z) |> e_add_nested("label", show, fontStyle)
```

---

**pie_action**  
Select & Unselect Pie

---

**Description**

Actions related to `e_pie`.

**Usage**

```r
e_pie_select(e, ..., btn = NULL)
e_pie_unselect(e, ..., btn = NULL)
```
Arguments

- `e` An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `...` Any options, see official documentation
- `btn` A e_button id.

Examples

```
mtcars |> head() |> tibble::rownames_to_column("model") |> e_charts(model) |> e_pie(carb) |> e_pie_select(dataIndex = 0)
```

```
df <- data.frame(x = LETTERS[1:10], y = seq(1, 20, by = 2))
df |> e_charts(x) |> e_polar() |> e_angle_axis() |> e_radius_axis(x) |> e_bar(y, coord.system = "polar")
```
**renderEcharts4rBox**  
*Render box*

**Description**

Render an echarts4r box.

**Usage**

```r
renderEcharts4rBox(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

- `expr`: An expression that produces as `echarts4rBox`.
- `env`: The environment in which to evaluate `expr`.
- `quoted`: Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

---

**timeline-opt**  
*Timeline*

**Description**

Set timeline options

**Usage**

```r
e_timeline_opts(e, axis_type = "category", ...)  
e_timeline_serie(e, ..., index = 1)  
e_timeline_on_serie(e, ..., serie_index)
```

**Arguments**

- `e`: An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- `axis_type`: Type of axis, time, value, or category.
- `...`: Named options.
- `index`: The index of the element to apply options to, see examples.
- `serie_index`: The index of the serie to add elements to.
Functions

- `e_timeline_opts`: Pass general timeline options, see official documentation.
- `e_timeline_serie`: Pass options to each serie, each options must be a vector or list the same length as their are steps, see examples.
- `e_timeline_make`: Helper function that wraps your data and `e_timeline_serie` to dynamically add options to series.

Examples

```r
# general options
iris |> 
group_by(Species) |> 
e_charts(Sepal.Length, timeline = TRUE) |> 
e_line(Sepal.Width) |> 
e_timeline_opts( 
  autoPlay = TRUE, 
  rewind = TRUE 
)

# serie options
iris |> 
group_by(Species) |> 
e_charts(Sepal.Length, timeline = TRUE) |> 
e_line(Sepal.Width) |> 
e_timeline_serie( 
  title = list( 
    list(text = "setosa"), 
    list(text = "versicolor"), 
    list(text = "virginica") 
  ) 
)
```

tooltip_action

Show & Hide Tooltip

Description

Show or hide tooltip.

Usage

```r
e_showtip(e, ..., btn = NULL)
e_hidetip(e, ..., btn = NULL)
```
Arguments

- **e**  
  An echarts4r object as returned by `e_charts` or a proxy as returned by `echarts4rProxy`.
- **...**  
  Any options, see [official documentation](#).
- **btn**  
  A `e_button` id.

Note

The tooltip must be initialised with `e_tooltip` for this to work.

Examples

```r
cars |>  
  e_charts(dist) |>  
  e_scatter(speed) |>  
  e_tooltip() |>  
  e_hidetip(btn = "btn") |>  
  e_button("btn", "Hide tooltip")
```
Index

angle_axis, 4
band, 5
band2, 6
callbacks, 7
colorRampPalette, 35
connections, 8
corrMatOrder, 36
countrycode, 37
e_add (nesting), 146
e_add_nested (nesting), 146
e_angle_axis (angle_axis), 4
e_angle_axis_ (angle_axis), 4
e_animation, 13
e_append1_p, 11, 14
e_append1_p_ (e_append1_p), 14
e_append2_p, 11
e_append2_p_ (e_append1_p), 14
e_append2_p_ (e_append1_p), 14
e_arc_g (e_graphic_g), 59
e_area, 16, 142
e_area_ (e_area), 16
e_aria, 18
e_arrange (connections), 8
e_axis, 18, 50
e_axis_ (e_axis), 18
e_axis_3d, 20
e_axis_formatter, 19
e_axis_formatter (e_axis), 18
e_axis_labels, 22
e_axis_pointer, 22
e_axis_stagger, 23
e_band (band), 5
e_band2 (band2), 6
e_band2_ (band2), 6
e_band_ (band), 5
e_bar, 23, 44, 142
e_bar_ (e_bar), 23
e_bar_3d, 25, 142
e_bar_3d_ (e_bar_3d), 25
e_bezier_curve_g (e_graphic_g), 59
e_boxplot, 27, 142
e_boxplot_ (e_boxplot), 27
e_brush, 17, 24, 28, 31, 63, 74, 98, 110, 121
e_button, 29, 106, 137–139, 143, 145, 148, 151
e_calendar, 30
e_candle, 31, 142
e_candle_ (e_candle), 31
e_capture, 32
e_chart (init), 140
e_charts_ (init), 140
e_charts_ (init), 140
e_circle_g (e_graphic_g), 59
e_cloud, 33
e_cloud_ (e_cloud), 33
e_color, 34
e_color_range, 35
e_color_range_ (e_color_range), 35
e_common, 36
e_connect (connections), 8
e_connect_group (connections), 8
e_correlations, 36
e_country_names, 37, 54, 56, 86
e_country_names_ (e_country_names), 37
e_data (init), 140
e_datazoom, 37
e_density, 142
e_density (e_histogram), 67

152
e_density_ (e_histogram), 67
e_dims, 38
e_disconnect_group (connections), 8
e_dispatch_action_p, 11, 39
e_downplay [highlight_action], 139
e_downplay_p, 11
e_downplay_p (e_highlight_p), 66
e_draft, 40
e_draw_p, 40, 141
e_effect_scatter, 142
e_effect_scatter (e_scatter), 108
e_effect_scatter_ (e_scatter), 108
e_error_bar, 41
e_error_bar_ (e_error_bar), 41
e_execute, 11, 43
e_execute_p (e_execute), 43
e_facet, 43
e_flip_coords, 44
e_flow_gl, 45
e_flow_gl_ (e_flow_gl), 45
e_focus_adjacency, 11
e_focus_adjacency (graph_action), 138
e_focus_adjacency_p, 48
e_format_axis, 50
e_format_x_axis (e_format_axis), 50
e_format_y_axis (e_format_axis), 50
e_funnel, 51, 146
e_funnel_ (e_funnel), 51
e_gauge, 52, 142
e_gauge_ (e_gauge), 52
e_geo, 53
e_geo_3d, 54
e_geo_3d_ (e_geo_3d), 54
e_get_data, 55
e_get_zr, 56
e_glm, 142
e_glm (eglm), 82
e_globe, 56
e_graph, 57, 138
e_graph_edges (e_graph), 57
e_graph_gb (e_graph), 57
e_graph_nodes (e_graph), 57
e_graphic_gb, 59
e_grid, 61
e_grid_3d, 62
e_group (connections), 8
e_group_g (e_graphic_g), 59
e_heatmap, 36, 63, 142
e_heatmap_ (e_heatmap), 63
e_hide_grid_lines, 65
e_hide_loading (e_show_loading), 118
e_hidetip (tooltip_action), 150
e_hidetip_p, 11
e_hidetip_p (e_showtip_p), 116
e_highlight (highlight_action), 139
e_highlight_p, 11, 66
e_histogram, 67, 142
e_histogram_ (e_histogram), 67
e_image_g (e_graphic_g), 59
e_inspect, 69
e_labels, 70
e_leaflet, 71
e_leaflet_tile (e_leaflet), 71
e_legend, 72
e_legend_scroll (legend_action), 143
e_legend_select (legend_action), 143
e_legend_toggle_select (legend_action), 143
e_legend_unselect (legend_action), 143
e_line, 15, 69, 73, 83, 142
e_line_ (e_line), 73
e_line_3d, 15, 142
e_line_3d (e_lines_3d), 77
e_line_3d_ (e_lines_3d), 77
e_lines_g (e_graphic_g), 59
e_lines, 75, 142
e_lines_ (e_lines), 75
e_lines_3d, 77, 142
e_lines_3d_ (e_lines_3d), 77
e_lines_gl, 80
e_liquid, 81
e_liquid_ (e_liquid), 81
e_list, 81
e_llvm, 82, 142
e_locale, 84
e_locale_manual (e_locale), 84
e_loess, 142
e_loess (e_llvm), 82
e_map, 85, 89, 142
e_map_ (e_map), 85
e_map_3d, 142
e_map_3d (e_map), 85
e_map_3d (e_map), 85
e_map_3d_custom (e_map), 85
e_map_register, 88, 145
e_map_register_p (e_map_register), 88
e_map_register_ui (e_map_register), 88
e_map_register (map_actions), 145
e_map_toggle_select (map_actions), 145
e_map_unselect (map_actions), 145
e_mapbox (mapbox), 144
e_mark_area (e_mark_point), 91
e_mark_line (e_mark_point), 91
e_map_p, 89
e_mark_p_ (e_mark_p), 89
e_mark_point, 91
e_merge, 94
e_modularity, 58, 94
e_morph, 95
e_off (callbacks), 7
e_on (callbacks), 7
e_parallel, 96
e_parallel_ (e_parallel), 96
e_pictorial, 97, 142
e_pictorial_ (e_pictorial), 97
e_pie, 100, 130, 142, 147
e_pie_ (e_pie), 100
e_pie_select (pie_action), 147
e_pie_unselect (pie_action), 147
e_polar, 101
e_polygon_g (e_graphic_g), 59
e_polyline_g (e_graphic_g), 59
e_radar, 102
e_radar_ (e_radar), 102
e_radar_opts, 103, 103
e_radius_axis (radius_axis), 148
e_radius_axis_ (radius_axis), 148
e_rect_g (e_graphic_g), 59
e_remove, 104
e_remove_serie (e_remove), 104
e_remove_serie_p, 11
e_remove_serie_p (e_remove), 104
e_resize, 105
e_restore, 105
e_ring_g (e_graphic_g), 59
e_river, 106
e_river_ (e_river), 106
e_rm_axis (e_axis), 18
e_sankey, 107
e_sankey_ (e_sankey), 107
e_scale (e_scatter), 108
e_scatter, 15, 108, 142
e_scatter_ (e_scatter), 108
e_scatter_3d, 15, 112, 142
e_scatter_3d_ (e_scatter_3d), 112
e_scatter_GL, 115, 142
e_scatter_GL_ (e_scatter_GL), 115
e_sector_g (e_graphic_g), 59
e_show_loading, 118
e_tooltip (tooltip_action), 150
e_tooltip.p, 11, 116
e_single_axis, 120
e_step, 121, 142
e_step_ (e_step), 121
e_sunburst, 122
e_sunburst_ (e_sunburst), 122
e_surface, 125
e_surface_ (e_surface), 125
e_svg (e_map), 85
e_svg_ (e_map), 85
e_svg_register (e_map_register), 88
e_text_g (e_graphic_g), 59
e_text_style, 126
e_theme, 34, 126
e_theme_custom (e_theme), 126
e_theme_register (e_theme), 126
e_timeline_on_serie (timeline-opts), 149
e_timeline_opts (timeline-opts), 149
e_timeline_serie (timeline-opts), 149
e_title, 128
e_toolbox (e_toolbox_feature), 128
e_toolbox_feature, 38, 128
e_tooltip, 129, 151
e_tooltip_choro_formatter (e_tooltip), 129
 e_tooltip_item_formatter, 130, 131
e_tooltip_item_formatter (e_tooltip), 129
 e_tooltip_pie_formatter, 130
 e_tooltip_pie_formatter (e_tooltip), 129
 e_tooltip_pointer_formatter, 130, 131
e_tooltip_pointer_formatter (e_tooltip), 129
 e_tree, 131
e_tree_ (e_tree), 131
e_tree_map, 132
e_tree_map_ (e_tree_map), 132
e_unfocus_adjacency, 11
 e_unfocus_adjacency (graph_action), 138
e_unfocus_adjacency_p (e_focus_adjacency_p), 48
 e_utc, 134
INDEX

155

renderEcharts4rBox, 12, 149
timeline-opts, 149
toJSON, 69
tooltip_action, 150

e_visual_map, 36, 134  renderEcharts4rBox (echarts4r-shiny), 10
e_visual_map_ (e_visual_map), 134
e_visual_map_range, 136
e_x_axis (e_axis), 18
e_x_axis_ (e_axis), 18
e_x_axis_3d (e_axis_3d), 20
e_y_axis (e_axis), 18
e_y_axis_ (e_axis), 18
e_y_axis_3d (e_axis_3d), 20
e_z_axis (e_axis), 18
e_z_axis_ (e_axis), 18
e_z_axis_3d (e_axis_3d), 20
e_zoom, 137
echarts4r-shiny, 10
echarts4r_proxy (echarts4r-shiny), 10
echarts4rBox, 11, 149
echarts4rBoxOutput, 12, 13
echarts4rProxy (echarts4r-shiny), 10
echarts4rOutput (echarts4r-shiny), 10
echarts4rProxy, 4, 6–8, 14, 15, 17–19,
  21–25, 27–34, 36, 38–40, 42–46, 48,
  50, 52–56, 60–63, 65, 66, 68, 69,
  71–74, 76, 78, 80–84, 86, 90, 92, 94,
  97, 98, 101, 103–106, 108, 110, 113,
  115, 116, 118, 120, 121, 123,
  125–130, 132–135, 137–139,
  143–146, 148, 149, 151
echarts4rProxy (echarts4r-shiny), 10
echarts_from_json (e_inspect), 69
graph_action, 138
highlight_action, 139
hist, 69
init, 140
JS, 8
legend_action, 143
lm, 83
map_actions, 145
mapbox, 144
nesting, 146
pie_action, 147
radius_axis, 148
renderEcharts4r (echarts4r-shiny), 10