

Package ‘echarts4r’

October 4, 2021

Title Create Interactive Graphs with 'Echarts JavaScript' Version 5

Date 2021-10-04

Version 0.4.2

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 7.1.2

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

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

NeedsCompilation no

Author John Coene [aut, cre, cph],
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]

Maintainer John Coene <jcoenep@gmail.com>

Repository CRAN

Date/Publication 2021-10-04 18:10:07 UTC

R topics documented:

| | |
|---------------------|----|
| angle_axis | 4 |
| band | 5 |
| band2 | 6 |
| callbacks | 7 |
| connections | 8 |
| echarts4r-shiny | 10 |
| echarts4rBox | 11 |
| echarts4rBoxOutput | 13 |
| e_animation | 13 |
| e_append1_p | 14 |
| e_area | 16 |
| e_aria | 18 |
| e_axis | 18 |
| e_axis_3d | 20 |
| e_axis_labels | 22 |
| e_axis_pointer | 22 |
| e_axis_stagger | 23 |
| e_bar | 23 |
| e_bar_3d | 25 |
| e_boxplot | 27 |
| e_brush | 28 |
| e_button | 29 |
| e_calendar | 30 |
| e_candle | 30 |
| e_capture | 32 |
| e_cloud | 33 |
| e_color | 34 |
| e_color_range | 35 |
| e_common | 36 |
| e_correlations | 36 |
| e_country_names | 37 |
| e_datazoom | 37 |
| e_dims | 38 |
| e_dispatch_action_p | 39 |
| e_draft | 40 |
| e_draw_p | 40 |
| e_error_bar | 41 |
| e_execute | 43 |
| e_flip_coords | 43 |
| e_flow_gl | 44 |
| e_focus_adjacency_p | 47 |
| e_format_axis | 48 |
| e_funnel | 49 |
| e_gauge | 51 |
| e_geo | 52 |
| e_geo_3d | 53 |

| | |
|-----------------------------|-----|
| e_get_data | 54 |
| e_globe | 55 |
| e_graph | 56 |
| e_graphic_g | 58 |
| e_grid | 60 |
| e_grid_3d | 60 |
| e_heatmap | 61 |
| e_hide_grid_lines | 64 |
| e_highlight_p | 65 |
| e_histogram | 66 |
| e_inspect | 68 |
| e_labels | 69 |
| e_leaflet | 70 |
| e_legend | 71 |
| e_line | 72 |
| e_lines | 74 |
| e_lines_3d | 76 |
| e_lines_gl | 79 |
| e_liquid | 80 |
| e_list | 81 |
| e_lm | 82 |
| e_locale | 83 |
| e_map | 84 |
| e_map_register | 87 |
| e_mark_p | 88 |
| e_mark_point | 91 |
| e_merge | 93 |
| e_modularity | 93 |
| e_parallel | 94 |
| e_pictorial | 95 |
| e_pie | 98 |
| e_polar | 99 |
| e_radar | 100 |
| e_radar_opts | 101 |
| e_remove | 102 |
| e_resize | 103 |
| e_restore | 103 |
| e_river | 104 |
| e_sankey | 105 |
| e_scatter | 106 |
| e_scatter_3d | 110 |
| e_scatter_gl | 112 |
| e_showtip_p | 114 |
| e_show_loading | 115 |
| e_single_axis | 117 |
| e_step | 118 |
| e_sunburst | 120 |
| e_surface | 122 |

| | |
|------------------------------|-----|
| e_text_style | 123 |
| e_theme | 124 |
| e_title | 126 |
| e_toolbox_feature | 127 |
| e_tooltip | 128 |
| e_tree | 130 |
| e_treemap | 131 |
| e_utc | 133 |
| e_visual_map | 133 |
| e_visual_map_range | 135 |
| e_zoom | 136 |
| graph_action | 137 |
| highlight_action | 138 |
| init | 138 |
| legend_action | 141 |
| mapbox | 142 |
| map_actions | 143 |
| nesting | 144 |
| pie_action | 146 |
| radius_axis | 147 |
| renderEcharts4rBox | 147 |
| timeline-opts | 148 |
| tooltip_action | 149 |

Index 150

| | |
|------------|-------------------|
| angle_axis | <i>Angle axis</i> |
|------------|-------------------|

Description

Customise angle axis.

Usage

```
e_angle_axis(e, serie, show = TRUE, ...)
```

```
e_angle_axis_(e, serie = NULL, show = TRUE, ...)
```

Arguments

| | |
|-------|--|
| e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| serie | Serie to use as axis labels. |
| show | Whether to display the axis. |
| ... | Any other option to pass, check See Also section. |

See Also

[Additional arguments](#)

Examples

```
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

```
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

| | |
|------------------------|---|
| <code>e</code> | An echarts4r object as returned by <code>e_charts</code> or a proxy as returned by <code>echarts4rProxy</code> . |
| <code>min, max</code> | series. |
| <code>stack</code> | Name of stack. |
| <code>symbol</code> | Whether to show symbols on lower and upper band lines. |
| <code>areaStyle</code> | The style of lower and upper bands, i.e.: color. |
| <code>legend</code> | Whether to show min and max in legend. |
| <code>...</code> | 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)

```

band2

Area bands

Description

Add area bands

Usage

```

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

e_band2_(
  e,

```

```

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

```

Arguments

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

```
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

```
echarts4rOutput(outputId, width = "100%", height = "400px")
```

```
renderEcharts4r(expr, env = parent.frame(), quoted = FALSE)
```

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

```
echarts4r_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 o 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`

echarts4rBox

Box

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

| | |
|-------------------------------|--|
| <code>data</code> | A dataframe containing data to plot. |
| <code>x, y</code> | Bare column name of variables to draw. |
| <code>text, subtext</code> | Title and subtitle of box. |
| <code>type</code> | Chart type to draw. |
| <code>...</code> | Additional arguments to pass to the serie. |
| <code>color</code> | Color of chart in box. |
| <code>text_color</code> | Color of text. |
| <code>background_color</code> | Color of box. |
| <code>step</code> | Step method, only used if <code>type = "step"</code> . |
| <code>title_args</code> | Additional arguments to add to the title. |
| <code>tooltip</code> | 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

```
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

```

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

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

e_append1_p

Append Proxy

Description

Append data dynamically.

Usage

```
e_append1_p(proxy, series_index = NULL, data, x, y)

e_append1_p(proxy, series_index = NULL, data, x, y)

e_append2_p(
  proxy,
  series_index = NULL,
  data,
  x,
  y,
  z,
  scale = NULL,
```

```

    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. |
| 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))
  })
}

```

```

  })

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

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

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

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

shinyApp(ui, server)

## End(Not run)

```

e_area

Area

Description

Add area serie.

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. |
| y_index | Indexes of x and y axis. |
| x_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

```
C02 |>
  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 | <i>Aria</i> |
|--------|-------------|

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 | <i>Axis</i> |
|--------|-------------|

Description

Customise axis.

Usage

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

```

)

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 <code>Sys.getlocale</code> . |
| currency | Currency to to display. |

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

```
# 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

Axis 3D

Description

Customise 3D axis.

Usage

```
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 <code>e_charts</code> or a proxy as returned by <code>echarts4rProxy</code> . |
| 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

```
# 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 | <i>Axis Labels</i> |
|---------------|--------------------|

Description

Convenience function to add axis labels.

Usage

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

Arguments

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

Examples

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

| | |
|----------------|---------------------|
| e_axis_pointer | <i>Axis pointer</i> |
|----------------|---------------------|

Description

Customise axis pointer.

Usage

```
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 | <i>Stagger Axis Labels</i> |
|----------------|----------------------------|

Description

Stagger axis labels.

Usage

```
e_axis_stagger(e)
```

Arguments

e An echarts4r object as returned by [e_charts](#) or a proxy as returned by [echarts4rProxy](#).

Examples

```
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 | <i>Bar and Line chart</i> |
|-------|---------------------------|

Description

Add bar serie.

Usage

```
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

```

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")

```

| | |
|----------|---------------|
| e_bar_3d | <i>Bar 3D</i> |
|----------|---------------|

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 | 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

```
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 | Indexes of x and y axis. |
| 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](#)

Examples

```

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")

```

`e_button`*Button*

Description

Add a button to your visualisation.

Usage

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

Arguments

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

Examples

```

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

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

```
# year
mtcars |>
  e_charts() |>
  e_calendar(range = "2017")

# month
mtcars |>
  e_charts() |>
  e_calendar(range = "2018-01")

# range
mtcars |>
  e_charts() |>
  e_calendar(range = c("2018-01", "2018-07"))
```

e_candle

Candlestick

Description

Add a candlestick chart.

Usage

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

```
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

```
date <- c(
  "2017-01-01",
  "2017-01-02",
  "2017-01-03",
  "2017-01-04",
  "2017-03-05",
  "2017-01-06",
  "2017-01-07"
)

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)
)
```

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

| | |
|-----------|----------------------|
| e_capture | <i>Capture event</i> |
|-----------|----------------------|

Description

Add an event capture.

Usage

```
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

```
## 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

```

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

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

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

## S3 method for class 'echarts4rProxy'
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](#)

Examples

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

| | |
|---------------|--------------------|
| e_color_range | <i>Color range</i> |
|---------------|--------------------|

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 | <i>General options</i> |
|----------|------------------------|

Description

General options

Usage

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

Arguments

| | |
|-------------|--------------|
| font_family | Font family. |
| theme | A theme. |

| | |
|----------------|--------------------|
| e_correlations | <i>Correlation</i> |
|----------------|--------------------|

Description

Correlation

Usage

```
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

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

| | |
|-----------------|----------------------|
| e_country_names | <i>Country names</i> |
|-----------------|----------------------|

Description

Convert country names to echarts format.

Usage

```
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

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

| | |
|------------|------------------|
| e_datazoom | <i>Data zoom</i> |
|------------|------------------|

Description

Add data zoom.

Usage

```
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 | Indexes of x and y axis. |
| 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

```
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

Dimensions

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

```
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. |

e_dispatch_action_p *Dispatch Action*

Description

Create your own proxies, essentially a wrapper around the [action API](#).

Usage

```
e_dispatch_action_p(proxy, type, ...)
```

Arguments

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

Examples

```
## Not run:

library(shiny)

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

server <- function(input, output, session) {
  output$chart <- renderEcharts4r({
    cars |>
      e_charts(speed) |>
      e_scatter(dist) |>
      e_datazoom()
  })

  observe({
    req(input$zoom)

    echarts4rProxy("chart") |>
      e_dispatch_action_p("dataZoom", startValue = 1, endValue = 10)
  })
}

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

```
## End(Not run)
```

| | |
|---------|--------------|
| e_draft | <i>Draft</i> |
|---------|--------------|

Description

Add a draft watermark to your graph.

Usage

```
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

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

| | |
|----------|-------------|
| e_draw_p | <i>Draw</i> |
|----------|-------------|

Description

Draw the chart.

Usage

```
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",
  ...
)
```

```

)

e_error_bar_(
  e,
  lower,
  upper,
  name = NULL,
  legend = FALSE,
  y_index = 0,
  x_index = 0,
  coord_system = "cartesian2d",
  itemStyle = list(borderWidth = 1.5),
  renderer = "renderErrorBar2",
  ...
)

```

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. |
| y_index | Indexes of x and y axis. |
| x_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, 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_flip_coords

Flip coordinates

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

```
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

```
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",
        "#ffffbf",
        "#fee090",
        "#fdae61",
        "#f46d43",
        "#d73027",
        "#a50026"
      )
    )
  )
```

```

) |>
e_x_axis(
  splitLine = list(show = FALSE)
) |>
e_y_axis(
  splitLine = list(show = FALSE)
)

# 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",
        "#ffffbf",
        "#fee090",
        "#fdae61",
        "#f46d43",
        "#d73027",
        "#a50026"
      )
    )
  ) |>
  e_x_axis(show = FALSE) |>

```

```
e_y_axis(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

```
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:

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)

```


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

```
# 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")
```

e_funnel

Funnel

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 <code>e_charts</code> . |
| 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

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

funnel |>
```

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

e_gauge

Gauge

Description

Plot a gauge.

Usage

```
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

```
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")
  )
```

e_geo

Geo

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))
  )
```

e_geo_3d

Geo 3D

Description

Initialise geo 3D.

Usage

```
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

```
choropleth <- data.frame(
  countries = c(
    "France",
    "Brazil",
    "China",
    "Russia",
    "Canada",
    "India",
    "United States",
    "Argentina",
    "Australia"
```

```
),
height = runif(9, 1, 5),
color = c(
  "#F7FBFF",
  "#DEEBF7",
  "#C6DBEF",
  "#9ECAE1",
  "#6BAED6",
  "#4292C6",
  "#2171B5",
  "#08519C",
  "#08306B"
)
)

choropleth |>
  e_charts(countries) |>
  e_geo_3d(height, color)
```

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_globe | <i>Globe</i> |
|---------|--------------|

Description

Add globe.

Usage

```
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

```
## 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_globe(
    displacementScale = 0.04
  ) |>
  e_bar_3d(lat, value, "globe") |>
  e_visual_map(show = FALSE)

## End(Not run)
```

e_graph

*Graph***Description**

Create a graph.

Usage

```
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

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

# Use graphGL for larger networks
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
)

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, ...)

e_arc_g(e, ...)

e_polygon_g(e, ...)

e_polyline_g(e, ...)

e_line_g(e, ...)

e_bezier_curve_g(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.

Functions

- e_graphic_g to initialise graphics, entirely optional.
- e_group_g to create group, the children of which will share attributes.
- e_image_g to a png or jpg image.
- e_text_g to add text.
- e_rect_g to add a rectangle.
- e_circle_g to add a circle.
- e_ring_g to add a ring.
- e_sector_g
- e_arc_g to create an arc.
- e_polygon_g to create a polygon.
- e_polyline_g to create a polyline.
- e_line_g to draw a line.
- e_bezier_curve_g to draw a quadratic bezier curve or cubic bezier curve.

Note

Some elements, i.e.: 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

[official documentation](#)

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
    )
  )
```

 e_grid

Grid

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 echarts4rProxy . |
| index | Index of axis to customise. |
| ... | Any other option to pass, check See Also section. |

See Also

[Additional arguments](#)

Examples

```
USArrests |>
  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%")
```

 e_grid_3d

Grid

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

```
e_heatmap(
  e,
  y,
```

```

    z,
    bind,
    name = NULL,
    coord_system = "cartesian2d",
    rm_x = TRUE,
    rm_y = TRUE,
    calendar = NULL,
    ...
)

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 serie. |
| 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

```

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_heatmap(
    lat,
    mag,
    coord_system = "geo",
    blurSize = 5,
    pointSize = 3
  ) |>
  e_visual_map(mag)

# timeline
library(dplyr)

axis <- LETTERS[1:10]
```

```
df <- expand.grid(axis, axis)

bind_rows(df, df) |>
  mutate(
    values = runif(n(), 1, 10),
    grp = c(
      rep("A", 100),
      rep("B", 100)
    )
  ) |>
  group_by(grp) |>
  e_charts(Var1, timeline = TRUE) |>
  e_heatmap(Var2, values) |>
  e_visual_map(values)
```

e_hide_grid_lines *'Hide Grid Lines*

Description

A convenience function to easily hide grid lines.

Usage

```
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

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

| | |
|---------------|---------------------------------------|
| e_highlight_p | <i>Highlight & Downplay Proxy</i> |
|---------------|---------------------------------------|

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_charts(mpg) |>
    e_line(displ) |>
    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 = "displ")
})

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

```

e_histogram(
  e,
  serie,
  breaks = "Sturges",

```

```
    name = NULL,  
    legend = TRUE,  
    bar_width = "99%",  
    x_index = 0,  
    y_index = 0,  
    ...  
  )  
  
  e_density(  
    e,  
    serie,  
    breaks = "Sturges",  
    name = NULL,  
    legend = TRUE,  
    x_index = 0,  
    y_index = 0,  
    smooth = TRUE,  
    ...  
  )  
  
  e_histogram_(  
    e,  
    serie,  
    breaks = "Sturges",  
    name = NULL,  
    legend = TRUE,  
    bar_width = "90%",  
    x_index = 0,  
    y_index = 0,  
    ...  
  )  
  
  e_density_(  
    e,  
    serie,  
    breaks = "Sturges",  
    name = NULL,  
    legend = TRUE,  
    x_index = 0,  
    y_index = 0,  
    smooth = 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.

| | |
|-----------|---|
| breaks | Passed to hist . |
| name | name of the serie. |
| legend | Whether to add serie to legend. |
| bar_width | Width of bars. |
| x_index | Indexes of x and y axis. |
| 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

```
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

```
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

```
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 <code>e_charts</code> or a proxy as returned by <code>echarts4rProxy</code> . |
| roam | Whether to allow the user to roam. |
| ... | Any other option to pass, check See Also section. |
| template | urlTemplate, should not be changed. |
| options | List of options, including attribution and label. |

Note

Will not render in the RStudio, open in browser.

Examples

```
## 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

```
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

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

# with legend
e

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

# with icon
# path is taken from http://svgicons.sparkk.fr/
path <- paste0(
  "path://M11.344,5.71c0-0.73,0.074-1.122,1.199-1.122",
  "h1.502V1.871h-2.404c-2.886,0-3.903,1.36-3.903,3.646",
  "v1.765h-1.8V10h1.8v8.128h3.601V10h2.403l0.32-2.718h",
  "-2.724L11.344,5.71z"
)

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 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. |
| y_index | Indexes of x and y axis. |
| x_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) |>
  e_tooltip(trigger = "axis")

# 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

```

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,
  ...
)

```

```

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

```
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
      )
    }
  ")
)

# timeline
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"
  )

```

e_lines_3d

Lines 3D

Description

Add 3D lines.

Usage

```

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,
  ...
)

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

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

# Lines 3D
# Globe
# get tetures: echarts4r-assets.john-coene.com
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

```
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).

e_liquid *Liquid fill*

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")
```

| | |
|--------|------|
| e_list | List |
|--------|------|

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 <code>e_charts</code> or a proxy as returned by <code>echarts4rProxy</code> . |
| list | A list of options passed to <code>setOptions</code> . |
| append | if TRUE the list is appended to the options, otherwise it <i>overwrites</i> everything. |

Examples

```
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

```
e_lm(  
  e,  
  formula,  
  name = NULL,  
  legend = TRUE,  
  symbol = "none",  
  smooth = TRUE,  
  model_args = list(),  
  ...  
)
```

```
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(),  
  ...  
)
```

Arguments

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

| | |
|------------|--|
| formula | formula to pass to lm . |
| name | name of the serie. |
| legend | Whether to add serie to legend. |
| symbol | Symbol to use in e_line . |
| smooth | Whether to smooth the line. |
| model_args | Arguments to pass to the underlying model. |
| ... | Additional arguments to pass to e_line . |
| x_index | Indexes of x and y axis. |
| y_index | Indexes of x and y axis. |

Examples

```
iris |>
  group_by(Species) |>
  e_charts(Sepal.Length) |>
  e_scatter(Sepal.Width) |>
  e_lm(Sepal.Width ~ Sepal.Length) |>
  e_x_axis(min = 4)

mtcars |>
  e_charts(displ) |>
  e_scatter(mpg, qsec) |>
  e_loess(mpg ~ displ, smooth = TRUE, showSymbol = FALSE)

# timeline
iris |>
  group_by(Species) |>
  e_charts(Sepal.Length, timeline = TRUE) |>
  e_scatter(Sepal.Width) |>
  e_lm(Sepal.Width ~ Sepal.Length) |>
  e_x_axis(min = 4, max = 8) |>
  e_y_axis(max = 5)
```

| | |
|----------|---------------|
| e_locale | <i>Locale</i> |
|----------|---------------|

Description

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

Usage

```
e_locale(e, locale)
```

Arguments

| | |
|--------|--|
| e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| locale | Locale to set to. |

Locales

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

Examples

```
# top right corner zoom is in
# Portuguese
cars |>
  e_charts(speed) |>
  e_scatter(dist) |>
  e_datazoom() |>
  e_locale("PT")
```

e_map

Choropleth

Description

Draw maps.

Usage

```
e_map(e, serie, map = "world", name = NULL, rm_x = TRUE, rm_y = TRUE, ...)
```

```
e_map_(
  e,
  serie = NULL,
  map = "world",
  name = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
```

```
e_svg(e, serie, map = "world", name = NULL, rm_x = TRUE, rm_y = TRUE, ...)
```

```
e_svg_(
  e,
  serie = NULL,
  map = "world",
  name = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
```

```
e_map_3d(
  e,
  serie,
  map = "world",
  name = NULL,
  coord_system = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  ...
)
```

```
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",
    "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 . |
| 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

```
## 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

```
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(
        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)
}

```

| | |
|--------------|-------------------|
| e_mark_point | <i>Mark point</i> |
|--------------|-------------------|

Description

Mark points and lines.

Usage

```
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
)
```

Arguments

| | |
|----------------|--|
| e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| serie | Serie or vector of series to mark on, defaults to all series. |
| data | Placement of point, line or area. |
| ... | Any other option to pass, check See Also section. |
| title | A convenience argument to easily set label, see details. |
| title_position | Position of title. |

Details

To set a label you need to either use the `title` argument or pass a list specifying the label formatter. `label = list(formatter = "label")`. The former is more convenient but more limited, e.g.: you cannot specify the placement of the label. When the `e_mark` series function is used with `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

```
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 = 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

```
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

```
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

```
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_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))
```

e_pictorial

Pictorial

Description

Pictorial bar chart is a type of bar chart that customized 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. |
| y_index | Indexes of x and y axis. |
| x_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

```
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

| | |
|------------|--|
| 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

```
mtcars |>
  head() |>
  tibble::rownames_to_column("model") |>
  e_charts(model) |>
  e_pie(carb)

# timeline
df <- data.frame(
  grp = c("A", "A", "A", "B", "B", "B"),
  labels = rep(LETTERS[1:3], 2),
  values = runif(6, 1, 5)
)

df |>
  group_by(grp) |>
  e_charts(labels, timeline = TRUE) |>
  e_pie(values)
```

e_polar

Polar

Description

Customise polar coordinates.

Usage

```
e_polar(e, show = TRUE, ...)
```

Arguments

| | |
|------|--|
| e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| show | Whether to display the axis. |
| ... | Any other option to pass, check See Also section. |

See Also

[Additional arguments](#)

Examples

```
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

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

```
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

```
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")
```

e_radar_opts

Radar axis

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")
```

e_remove

Remove Serie

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)
  )

  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**Resize*

Description

Force resize the chart.

Usage

```
e_resize(proxy)
```

Arguments

proxy An echarts4r proxy as returned by [echarts4rProxy](#).

*e_restore**Restore Toolbox*

Description

Restore Toolbox.

Usage

```
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")

```

| | |
|---------|--------------|
| e_river | <i>River</i> |
|---------|--------------|

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,  
  ...  
)
```

Arguments

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

See Also

[Additional arguments](#)

Examples

```
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

```
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 <i>not</i> 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. |
| y_index | Indexes of x and y axis. |
| x_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

```
# 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

```
e_scatter_3d(
  e,
  y,
  z,
  color,
  size,
  bind,
  coord_system = "cartesian3D",
  name = NULL,
  rm_x = TRUE,
  rm_y = TRUE,
  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

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

```
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

```
e_showtip_p(proxy, ...)
```

```
e_hidetip_p(proxy)
```

Arguments

proxy An echarts4r proxy as returned by [echarts4rProxy](#).

... Any other option, see [showTip](#).

Examples

```
## 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(displacement, 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)
```

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:

# 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

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

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

e_step

Step

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. |
| y_index | Indexes of x and y axis. |
| x_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,
  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. |

Details

Charts `e_sunburst`, `e_treemap` and `e_tree` require hierarchical input data. Such structure could be represented thru json lists or nested tibbles (`data.frame`). Input data may contain styles, see `itemStyle` in examples `json1` and `df` below. The number of lists in the `styles` parameter should match the number of elements in `names` and/or `levels`. If both `names` and `levels` are present, name styles will take precedence over level styles. Multiple names may have the same style, see `c('land', 'river')` below. Multiple levels may have the same style, see `c(3,4)` below. `styles` lists contain items such as `color`, or `borderColor` as specified in the [official documentation](#).

See Also

[Additional arguments](#)

Examples

```
# json list hierarchical data representation
json1 <- 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}
            ]},
          {"name": "kelp", "value": 5}
        ]}
    ]
  },
  {"name": "mars", "value": 30,
    "children": [
      {"name": "crater", "value": 20},
      {"name": "valley", "value": 20}
    ]},
  {"name": "venus", "value": 40, "itemStyle": {"color": "blue"} }
]', simplifyDataFrame = FALSE)

json1 |>
  e_charts() |>
  e_sunburst() # demo

# tibble hierarchical data representation
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() |>
  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

```
e_surface(e, y, z, bind, name = NULL, rm_x = TRUE, rm_y = TRUE, ...)
```

```
e_surface_(e, y, z, bind = 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](#).

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

```
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 | <i>Text style</i> |
|--------------|-------------------|

Description

Define global font style.

Usage

```
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

```
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)

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.

Theme names

- default
- dark
- vintage
- westeros
- essos
- wonderland
- walden
- chalk
- infographic
- macarons
- roma
- shine
- purple-passion
- halloween
- auritus
- azul
- bee-inspired
- blue
- caravan
- carp
- cool
- dark-blue
- dark-bold
- dark-digerati
- dark-fresh-cut
- dark-mushroom
- eduardo
- forest
- fresh-cut
- fruit
- gray
- green
- helianthus
- inspired
- jazz
- london

- macarons
- macarons2
- mint
- red
- red-velvet
- royal
- sakura
- tech-blue

See Also

[create your own theme.](#)

Examples

```
mtcars |>
  e_charts(mpg) |>
  e_line(displ) |>
  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

```
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

```
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 | <i>Toolbox</i> |
|-------------------|----------------|

Description

Add toolbox interface.

Usage

```
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

```

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")
  )

```

`e_tooltip`*Tooltip*

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 <code>item</code> or <code>item</code> . |
| 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 <code>decimal</code> , <code>percent</code> , or <code>currency</code> . |
| digits | Number of decimals. |
| locale | Locale, if NULL then it is inferred from <code>Sys.getlocale</code> . |
| 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

```

# basic
USArrests |>
  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"
    )
  )
)

```

e_tree

Tree

Description

Build a tree.

Usage

```
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

```

library(dplyr)
df <- tibble(
  name = "earth",
  # 1st level
  children = list(
    tibble(
      name = c("land", "ocean"),
      # 2nd level
      children = list(
        tibble(name = c("forest", "river")),

```


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
      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 | <i>Use UTC</i> |
|-------|----------------|

Description

Use UTC

Usage

```
e_utc(e)
```

Arguments

e An echarts4r object as returned by [e_charts](#) or a proxy as returned by [echarts4rProxy](#).

| | |
|--------------|-------------------|
| e_visual_map | <i>Visual Map</i> |
|--------------|-------------------|

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*

Description

Selects data range of visual mapping.

Usage

```
e_visual_map_range(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

```

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

```
e_zoom(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

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

| | |
|--------------|------------------------|
| graph_action | <i>Nodes Adjacency</i> |
|--------------|------------------------|

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
)

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 | <i>Highlight & Downplay</i> |
|------------------|---------------------------------|

Description

Highlight series

Usage

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

```
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

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

| | |
|------|-------------------|
| init | <i>Initialise</i> |
|------|-------------------|

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,
  elementId = NULL,
  dispose = TRUE,
  draw = TRUE,
  renderer = "canvas",
  timeline = FALSE,
  ...,
  reorder = TRUE
)
```

```

)

e_chart(
  data,
  x,
  width = NULL,
  height = NULL,
  elementId = 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. |
| elementId | 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 <code>echarts4r</code> as returned by <code>e_charts</code> . |

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_lm](#)
- [e_loess](#)
- [e_glm](#)
- [e_density](#)
- [e_pictorial](#)
- [e_boxplot](#)
- [e_map](#)
- [e_map_3d](#)
- [e_line_3d](#)
- [e_gauge](#)

Examples

```
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 = "blue")
```

legend_action

Legend

Description

Legend

Usage

```
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

| | |
|---------------------------|--|
| <code>e</code> | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| <code>name</code> | Legend name. |
| <code>btn</code> | A e_button id. |
| <code>scroll_index</code> | Controlle the scrolling of legend when type = "scroll" in e_legend. |
| <code>legend_id</code> | Id of legend. |

Examples

```
e <- C02 |>
  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")
```

 mapbox

Mapbox

Description

Use mapbox.

Usage

```
e_mapbox(e, token, ...)
```

Arguments

| | |
|--------------------|--|
| <code>e</code> | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| <code>token</code> | Your mapbox token from mapbox . |
| <code>...</code> | Any option. |

Note

Mapbox may not work properly in the RSudio console.

See Also

[Official documentation](#), [mapbox documentation](#)

Examples

```
## 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

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")

```

nesting*Add nested data*

Description

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

Usage

```
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

| | |
|---------------------|---|
| <code>e</code> | An <code>echarts4r</code> object as returned by <code>e_charts</code> or a proxy as returned by <code>echarts4rProxy</code> . |
| <code>param</code> | The nested parameter to add data to. |
| <code>...</code> | Any other option to pass, check See Also section. |
| <code>.serie</code> | 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

```
# 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),
  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 | <i>Select & Unselect Pie</i> |
|------------|----------------------------------|

Description

Actions related to [e_pie](#).

Usage

```
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)
```

| | |
|-------------|--------------------|
| radius_axis | <i>Radius axis</i> |
|-------------|--------------------|

Description

Customise radius axis.

Usage

```
e_radius_axis(e, serie, show = TRUE, ...)
```

```
e_radius_axis_(e, serie = NULL, show = TRUE, ...)
```

Arguments

| | |
|-------|--|
| e | An echarts4r object as returned by e_charts or a proxy as returned by echarts4rProxy . |
| serie | Serie to use as axis labels. |
| show | Whether to display the axis. |
| ... | Any other option to pass, check See Also section. |

See Also

[Additional arguments](#)

Examples

```
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 | <i>Render box</i> |
|--------------------|-------------------|

Description

Render an echarts4r box.

Usage

```
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-opts | <i>Timeline</i> |
|---------------|-----------------|

Description

Set timeline options

Usage

```
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

```
# 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

```
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

```
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), 144
e_add_nested (nesting), 144
e_add_unnested (nesting), 144
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), 58
e_area, 16, 140
e_area_ (e_area), 16
e_aria, 18
e_arrange (connections), 8
e_axis, 18, 49
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, 140
e_bar_ (e_bar), 23
e_bar_3d, 25, 141
e_bar_3d_ (e_bar_3d), 25
e_bezier_curve_g (e_graphic_g), 58
e_boxplot, 27, 141
e_boxplot_ (e_boxplot), 27
e_brush, 17, 24, 28, 31, 62, 73, 96, 108, 119
e_button, 29, 103, 135–138, 142, 143, 146, 149
e_calendar, 30
e_candle, 30, 140
e_candle_ (e_candle), 30
e_capture, 32
e_chart (init), 138
e_charts, 4, 6–8, 14, 17–19, 21–25, 27–34, 36, 38, 40, 42, 43, 45, 49–55, 58, 60–62, 64, 67, 68, 70–73, 75, 77, 79–83, 85, 88, 89, 91, 94–96, 98–101, 103–105, 108, 111, 113, 116, 118–120, 123, 124, 126, 127, 129, 130, 132–138, 142–144, 146–149
e_charts (init), 138
e_charts_ (init), 138
e_circle_g (e_graphic_g), 58
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, 53, 55, 86
e_country_names_ (e_country_names), 37
e_data (init), 138
e_datazoom, 37
e_density, 141

- e_density (e_histogram), 66
- e_density_ (e_histogram), 66
- e_dims, 38
- e_disconnect_group (connections), 8
- e_dispatch_action_p, 11, 39
- e_downplay (highlight_action), 138
- e_downplay_p, 11
- e_downplay_p (e_highlight_p), 65
- e_draft, 40
- e_draw_p, 40, 140
- e_effect_scatter, 140
- e_effect_scatter (e_scatter), 106
- e_effect_scatter_ (e_scatter), 106
- e_error_bar, 41
- e_error_bar_ (e_error_bar), 41
- e_execute, 11, 43
- e_execute_p (e_execute), 43
- e_flip_coords, 43
- e_flow_gl, 44
- e_flow_gl_ (e_flow_gl), 44
- e_focus_adjacency, 11
- e_focus_adjacency (graph_action), 137
- e_focus_adjacency_p, 47
- e_format_axis, 48
- e_format_x_axis (e_format_axis), 48
- e_format_y_axis (e_format_axis), 48
- e_funnel, 49, 145
- e_funnel_ (e_funnel), 49
- e_gauge, 51, 141
- e_gauge_ (e_gauge), 51
- e_geo, 52
- e_geo_3d, 53
- e_geo_3d_ (e_geo_3d), 53
- e_get_data, 54
- e_glm, 141
- e_glm (e_lm), 82
- e_globe, 55
- e_graph, 56, 137
- e_graph_edges (e_graph), 56
- e_graph_gl (e_graph), 56
- e_graph_nodes (e_graph), 56
- e_graphic_g, 58
- e_grid, 60
- e_grid_3d, 60
- e_group (connections), 8
- e_group_g (e_graphic_g), 58
- e_heatmap, 36, 61, 141
- e_heatmap_ (e_heatmap), 61
- e_hide_grid_lines, 64
- e_hide_loading (e_show_loading), 116
- e_hidetip (tooltip_action), 149
- e_hidetip_p, 11
- e_hidetip_p (e_showtip_p), 114
- e_highlight (highlight_action), 138
- e_highlight_p, 11, 65
- e_histogram, 66, 141
- e_histogram_ (e_histogram), 66
- e_image_g (e_graphic_g), 58
- e_inspect, 68
- e_labels, 69
- e_leaflet, 70
- e_leaflet_tile (e_leaflet), 70
- e_legend, 71
- e_legend_scroll (legend_action), 141
- e_legend_select (legend_action), 141
- e_legend_toggle_select (legend_action), 141
- e_legend_unselect (legend_action), 141
- e_line, 15, 68, 72, 83, 140
- e_line_ (e_line), 72
- e_line_3d, 15, 141
- e_line_3d (e_lines_3d), 76
- e_line_3d_ (e_lines_3d), 76
- e_line_g (e_graphic_g), 58
- e_lines, 74, 141
- e_lines_ (e_lines), 74
- e_lines_3d, 76, 141
- e_lines_3d_ (e_lines_3d), 76
- e_lines_gl, 79
- e_liquid, 80
- e_liquid_ (e_liquid), 80
- e_list, 81
- e_lm, 82, 141
- e_locale, 83
- e_loess, 141
- e_loess (e_lm), 82
- e_map, 84, 88, 141
- e_map_ (e_map), 84
- e_map_3d, 141
- e_map_3d (e_map), 84
- e_map_3d_ (e_map), 84
- e_map_3d_custom (e_map), 84
- e_map_register, 87, 144
- e_map_register_p (e_map_register), 87
- e_map_register_ui (e_map_register), 87
- e_map_select (map_actions), 143

- e_map_toggle_select (map_actions), 143
- e_map_unselect (map_actions), 143
- e_mapbox (mapbox), 142
- e_mark_area (e_mark_point), 91
- e_mark_line (e_mark_point), 91
- e_mark_p, 88
- e_mark_p_ (e_mark_p), 88
- e_mark_point, 91
- e_merge, 93
- e_modularity, 57, 93
- e_off (callbacks), 7
- e_on (callbacks), 7
- e_parallel, 94
- e_parallel_ (e_parallel), 94
- e_pictorial, 95, 141
- e_pictorial_ (e_pictorial), 95
- e_pie, 98, 129, 141, 146
- e_pie_ (e_pie), 98
- e_pie_select (pie_action), 146
- e_pie_unselect (pie_action), 146
- e_polar, 99
- e_polygon_g (e_graphic_g), 58
- e_polyline_g (e_graphic_g), 58
- e_radar, 100
- e_radar_ (e_radar), 100
- e_radar_opts, 101, 101
- e_radius_axis (radius_axis), 147
- e_radius_axis_ (radius_axis), 147
- e_rect_g (e_graphic_g), 58
- e_remove, 102
- e_remove_serie (e_remove), 102
- e_remove_serie_p, 11
- e_remove_serie_p (e_remove), 102
- e_resize, 103
- e_restore, 103
- e_ring_g (e_graphic_g), 58
- e_river, 104
- e_river_ (e_river), 104
- e_rm_axis (e_axis), 18
- e_sankey, 105
- e_sankey_ (e_sankey), 105
- e_scale (e_scatter), 106
- e_scatter, 15, 106, 140
- e_scatter_ (e_scatter), 106
- e_scatter_3d, 15, 110, 141
- e_scatter_3d_ (e_scatter_3d), 110
- e_scatter_gl, 112, 141
- e_scatter_gl_ (e_scatter_gl), 112
- e_sector_g (e_graphic_g), 58
- e_show_loading, 115
- e_showtip (tooltip_action), 149
- e_showtip_p, 11, 114
- e_single_axis, 117
- e_step, 118, 140
- e_step_ (e_step), 118
- e_sunburst, 120
- e_sunburst_ (e_sunburst), 120
- e_surface, 122
- e_surface_ (e_surface), 122
- e_svg (e_map), 84
- e_svg_ (e_map), 84
- e_svg_register (e_map_register), 87
- e_text_g (e_graphic_g), 58
- e_text_style, 123
- e_theme, 34, 124
- e_theme_custom (e_theme), 124
- e_theme_register (e_theme), 124
- e_timeline_on_serie (timeline_opts), 148
- e_timeline_opts (timeline_opts), 148
- e_timeline_serie (timeline_opts), 148
- e_title, 126
- e_toolbox (e_toolbox_feature), 127
- e_toolbox_feature, 38, 127
- e_tooltip, 128, 149
- e_tooltip_choro_formatter (e_tooltip), 128
- e_tooltip_item_formatter, 129
- e_tooltip_item_formatter (e_tooltip), 128
- e_tooltip_pie_formatter, 129
- e_tooltip_pie_formatter (e_tooltip), 128
- e_tooltip_pointer_formatter, 129
- e_tooltip_pointer_formatter (e_tooltip), 128
- e_tree, 130
- e_tree_ (e_tree), 130
- e_treemap, 131
- e_treemap_ (e_treemap), 131
- e_unfocus_adjacency, 11
- e_unfocus_adjacency (graph_action), 137
- e_unfocus_adjacency_p (e_focus_adjacency_p), 47
- e_utc, 133
- e_visual_map, 36, 133
- e_visual_map_ (e_visual_map), 133
- e_visual_map_range, 135

`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`, 136
`echarts4r-shiny`, 10
`echarts4r_proxy` (`echarts4r-shiny`), 10
`echarts4rBox`, 11, 148
`echarts4rBoxOutput`, 12, 13
`echarts4rOutput` (`echarts4r-shiny`), 10
`echarts4rProxy`, 4, 6–8, 14, 15, 17–19,
21–25, 27–34, 36, 38–40, 42, 43, 45,
47, 49, 51–55, 58, 60–62, 64, 65, 67,
68, 70–73, 75, 77, 79–83, 85, 89, 91,
93–96, 98–105, 108, 111, 113, 114,
116, 118–120, 123, 124, 126, 127,
129, 130, 132–138, 142–144,
146–149
`echarts4rProxy` (`echarts4r-shiny`), 10
`echarts_from_json` (`e_inspect`), 68

`graph_action`, 137

`highlight_action`, 138
`hist`, 68

`init`, 138

JS, 8

`legend_action`, 141
`lm`, 83

`map_actions`, 143
`mapbox`, 142

`nesting`, 144

`pie_action`, 146

`radius_axis`, 147
`renderEcharts4r` (`echarts4r-shiny`), 10
`renderEcharts4rBox`, 12, 147

`timeline-opts`, 148
`toJSON`, 68
`tooltip_action`, 149