

# Package ‘echarts4r’

January 5, 2019

**Title** Create Interactive Graphs with 'Echarts JavaScript' Version 4

**Date** 2019-01-04

**Version** 0.2.1

**Description**

Easily create interactive charts by leveraging the 'Echarts Javascript' library which includes 34 chart types, themes, 'Shiny' proxies and animations.

**License** Apache License (>= 2.0)

**Encoding** UTF-8

**LazyData** true

**Imports** htmlwidgets, magrittr, dplyr, purrr, countrycode, d3r, broom, shiny, scales

**RoxygenNote** 6.1.1

**Suggests** data.tree, jsonlite, knitr, rmarkdown, quantmod, tibble, htmltools, tidyr, sp, raster, geojsonio, rmapshaper, corplot

**URL** <http://echarts4r.john-coene.com/>

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

**NeedsCompilation** no

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

**Maintainer** John Coene <jcoenep@gmail.com>

**Repository** CRAN

**Date/Publication** 2019-01-05 21:00:04 UTC

## R topics documented:

angle_axis . . . . .	4
band . . . . .	5
callbacks . . . . .	6
connections . . . . .	6
echarts4r-shiny . . . . .	8
e_add . . . . .	9

e_animation	10
e_append1_p	11
e_area	13
e_aria	14
e_axis	15
e_axis_3d	16
e_axis_pointer	17
e_bar	18
e_bar_3d	19
e_boxplot	21
e_brush	22
e_button	23
e_calendar	23
e_candle	24
e_capture	25
e_cloud	26
e_color	27
e_color_range	28
e_common	29
e_correlations	29
e_country_names	30
e_datazoom	30
e_dispatch_action_p	31
e_draft	32
e_error_bar	33
e_facet	34
e_flip_coords	35
e_flow_gl	35
e_focus_adjacency_p	37
e_format_axis	39
e_funnel	40
e_gauge	41
e_geo	42
e_geo_3d	43
e_get_data	44
e_globe	44
e_graph	45
e_graphic_g	47
e_grid	49
e_grid_3d	50
e_heatmap	51
e_highlight_p	53
e_histogram	54
e_labels	56
e_leaflet	56
e_legend	57
e_line	59
e_lines	60

e_lines_3d	61
e_lines_gl	64
e_liquid	64
e_list	65
e_lm	66
e_map	67
e_map_register	69
e_mark_point	70
e_modularity	71
e_parallel	72
e_pictorial	73
e_pie	75
e_polar	76
e_radar	77
e_radar_opts	78
e_restore	79
e_river	79
e_sankey	80
e_scatter	81
e_scatter_3d	84
e_scatter_gl	86
e_showtip_p	87
e_show_loading	89
e_single_axis	91
e_step	92
e_sunburst	93
e_surface	94
e_text_style	95
e_theme	96
e_title	97
e_toolbox_feature	98
e_tooltip	99
e_tree	100
e_treemap	101
e_utc	102
e_visual_map	102
e_visual_map_range	104
e_zoom	105
graph_action	105
highlight_action	106
init	107
legend_action	109
mapbox	110
map_actions	111
pie_action	112
radius_axis	112
timeline-opts	113
tooltip_action	114

---

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 <code>e_charts</code> .
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	<i>Confidence bands</i>
------	-------------------------

---

## 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

e	An echarts4r object as returned by <code>e_charts</code> .
min, max	series.
stack	Name of stack.
symbol	Whether to show symbols on lower and upper band lines.
areaStyle	The style of lower and upper bands, i.e.: color.
legend	Whether to show min and max in legend.
...	All options must be of vectors or lists of length 2 where the first argument is for the lower bound and the second for the upper bound, see examples.

## Examples

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

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

---

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](#).

`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

<code>e</code>	An echarts4r object as returned by <code>e_charts</code> .
<code>ids</code>	Scalar, vector or list of ids of chart to connect with.
<code>group</code>	Group name.
<code>...</code>	Any echarts objects.
<code>rows, cols</code>	Number of rows and columns.
<code>width</code>	Width of columns, one of <code>xs</code> , <code>md</code> , <code>lg</code> .
<code>title</code>	Title of charts.

## Value

`e_arrange`: in an interactive session, returns a [browsable](#), in rmarkdown returns a container ([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

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, session = shiny::getDefaultReactiveDomain())

```

## Arguments

<code>outputId</code>	output variable to read from.
<code>width, height</code>	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
<code>expr</code>	An expression that generates a echarts4r
<code>env</code>	The environment in which to evaluate <code>expr</code> .
<code>quoted</code>	Is <code>expr</code> a quoted expression (with <code>quote()</code> )? This is useful if you want to save an expression in a variable.
<code>id</code>	Target chart id.
<code>session</code>	Shiny session.



### 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

- [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\_add

*Add nested data*

---

### Description

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

### Usage

```
e_add(e, param, ...)
```

### Arguments

<code>e</code>	An <code>echarts4r</code> object as returned by <a href="#">e_charts</a> .
<code>param</code>	The nested parameter to add data to.
<code>...</code>	Any other option to pass, check <a href="#">See Also</a> section.

### 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.

**Examples**

```

# funnel can take nested itemStyle
# https://ecomfe.github.io/echarts-doc/public/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("itemStyle", color)

# Heatmap can take nested label
# https://ecomfe.github.io/echarts-doc/public/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(
    "label",
    show,
    fontStyle
  )

```

---

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 <code>e_charts</code> .
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 <a href="#">echarts4rProxy</a> .
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 <a href="#">e_scatter</a> .
symbol_size	Multiplier of scaling function as in <a href="#">e_scatter</a> .

**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 <a href="#">e_charts</a> .
serie	Column name of serie to plot.
bind	Binding between datasets, namely for use of <a href="#">e_brush</a> .

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

*Aria*

---

### Description

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

### Usage

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

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
show	Whether to show 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, axis = c("x", "y", "z"), index = 0, formatter = NULL, ...)
```

```
e_x_axis(e, index = 0, formatter = NULL, ...)
```

```
e_y_axis(e, index = 0, formatter = NULL, ...)
```

```
e_z_axis(e, index = 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 <a href="#">e_charts</a> .
axis	Axis to customise.
index	Index of axis to customise.
formatter	An axis formatter as returned by <a href="#">e_axis_formatter</a> .
...	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 display.

**Functions**

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

**See Also**

[Additional x arguments](#), [Additional y arguments](#)

**Examples**

```
# hide axis
USArrests %>%
  e_charts(Assault) %>%
  e_line(Murder, smooth = TRUE) %>%
  e_line(Rape, y.index = 1) %>% # add secondary axis
  e_y_axis(index = 1, show = FALSE) # hide secondary axis

# 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) %>%
  dplyr::mutate(State = row.names(.)) %>%
  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> .
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\_pointer

*Axis pointer*

---

### Description

Customise axis pointer.

**Usage**

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

**Arguments**

e                    An echarts4r object as returned by [e\\_charts](#).  
 ...                  Any other option to pass, check See Also section.

**See Also**

[Additional arguments](#)

---

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](#).  
 serie                Column name of serie to plot.  
 bind                Binding between datasets, namely for use of [e\\_brush](#).  
 name                name of the serie.  
 legend              Whether to add serie to legend.  
 x\_index, y\_index    Indexes of x and y axis.  
 coord\_system        Coordinate system to plot against.  
 ...                  Any other option to pass, check See Also section.

**See Also**

[Additional arguments](#)

**Examples**

```
library(dplyr)

mtcars %>%
  mutate(
    model = row.names(.),
    total = mpg + qsec
  ) %>%
  arrange(desc(total)) %>%
  e_charts(model) %>%
  e_bar(mpg, stack = "grp") %>%
  e_bar(qsec, stack = "grp")
```

---

e\_bar\_3d

*Bar 3D*


---

**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 <a href="#">e_charts</a> .
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://ecomfe.github.io/echarts-examples/",
              "public/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 <a href="#">e_charts</a> .
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, ...)
```

**Arguments**

**e** An echarts4r object as returned by `e_charts`.

**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.

**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	<i>Button</i>
----------	---------------

---

### Description

Add a button to your visualisation.

### Usage

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

### Arguments

e	An echarts4r object as returned by <code>e_charts</code> .
id	A valid CSS id.
...	Content of the button, compliant with <code>htmltools</code> .
position	Position of button, top or bottom.
tag	A Valid <a href="#">tags</a> 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	<i>Calendar</i>
------------	-----------------

---

### Description

Calendar

### Usage

```
e_calendar(e, range, ...)
```

**Arguments**

e An echarts4r object as returned by `e_charts`.  
 range Range of calendar format, string or vector.  
 ... Any other option to pass, check See Also section.

**See Also**

<https://ecomfe.github.io/echarts-doc/public/en/option.html#calendar>

**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`.  
 opening, closing, low, high  
 Stock prices.



bind	Binding between datasets, namely for use of <a href="#">e_brush</a> .
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

*Capture event*


---

**Description**

Add an event capture.

**Usage**

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

**Arguments**

e	An <code>echarts4r</code> object as returned by <a href="#">e_charts</a> .
event	An event name from the <a href="#">event documentation</a> .

**Details**

Many events can be capture, 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
  })
}

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 <a href="#">e_charts</a> .
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**

<https://github.com/ecomfe/echarts-wordcloud>

**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)
```

**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
color	Vector of colors.
background	Background color.

**See Also**

[e\\_theme](#), <https://ecomfe.github.io/echarts-doc/public/en/option.html#color>, <https://ecomfe.github.io/echarts-doc/public/en/option.html#backgroundColor>

## 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 <a href="#">colorRampPalette</a> .
...	Any other argument to pass to <a href="#">colorRampPalette</a> .

## 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 <a href="#">e_charts</a> .
order	Ordering method, passed to <a href="#">corrMatOrder</a> .
visual_map	Whether to add the visual map.
...	Any argument to pass to <a href="#">e_heatmap</a> and <a href="#">e_visual_map</a> .

**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 <a href="#">countrycode</a> origin parameter.
...	Any other parameter to pass to <a href="#">countrycode</a> .

**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 <a href="#">e_charts</a> .
x_index	Indexes of x and y axis.
y_index	Indexes of x and y axis.
toolbox	Whether to add the toolbox, <a href="#">e_toolbox_feature</a> , ( <code>e_toolbox_feature(e, "dataZoom")</code> ).
...	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\_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 <a href="#">echarts4rProxy</a> .
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)
  })

}

shinyApp(ui, server)

## End(Not run)
```

---

e\_draft

*Draft*

---

## 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](#).  
 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()
```

---

<code>e_error_bar</code>	<i>Error bar</i>
--------------------------	------------------

---

**Description**

Add error bars.

**Usage**

```
e_error_bar(e, lower, upper, name = NULL, legend = TRUE, y_index = 0,
  x_index = 0, coord_system = "cartesian2d", ...)
```

```
e_error_bar_(e, lower, upper, name = NULL, legend = TRUE,
  y_index = 0, x_index = 0, coord_system = "cartesian2d", ...)
```

**Arguments**

e An echarts4r object as returned by [e\\_charts](#).  
 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.

**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\_facet

*Facet*


---

**Description**

Create facets for multiple plots.

**Usage**

```
e_facet(e, rows = 1, cols = 1)
```

**Arguments**

**e** An echarts4r object as returned by `e_charts`.  
**rows, cols** Number of rows and columns.

**Details**

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

---

e_flip_coords	<i>Flip coordinates</i>
---------------	-------------------------

---

**Description**

Flip cartesian 2D coordinates.

**Usage**

```
e_flip_coords(e)
```

**Arguments**

`e` An `echarts4r` object as returned by `e_charts`.

**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	<i>Flow GL</i>
-----------	----------------

---

**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 <a href="#">e_charts</a> .
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(
    itemStyle = list(
      normal = list(
        areaColor = "#323c48",
        borderColor = "#111"
      )
    )
  ) %>%
  e_flow_gl(lat, slng, slat, color,
    coord_system = "geo",
    itemStyle = trans,
    particleSize = 2
  ) %>%
  e_visual_map(
    min = rng[1], max = rng[2], # 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\_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 <a href="#">echarts4rProxy</a> .
index	Index of node to focus on.
...	Any other options, see <a href="#">official documentation</a> 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(
        seriesIndex = 0,
        index = input$index
      )
  })

  observeEvent(input$unfocus, {
    echarts4rProxy("graph") %>%
      e_unfocus_adjacency(seriesIndex = 0)
  })
}

shinyApp(ui, server)

## End(Not run)

```

---

e\_format\_axis

*Formatters*


---

### Description

Simple formatters as helpers.

### Usage

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

```
e_format_x_axis(e, suffix = NULL, prefix = NULL, ...)
```

```
e_format_y_axis(e, suffix = NULL, prefix = NULL, ...)
```

### Arguments

**e** An echarts4r object as returned by [e\\_charts](#).

**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 <a href="#">e_charts</a> .
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 <a href="#">e_charts</a> .
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")
```

---

e\_geo

*Geo*

---

### Description

Initialise geo.

### Usage

```
e_geo(e, map = "world", ...)
```

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
map	Map type.
...	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,
    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 <a href="#">e_charts</a> .
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`.

**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`*Globe*

---

**Description**

Add globe.

**Usage**

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

**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
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://ecomfe.github.io/echarts-examples/",
              "public/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, ...)

e_graph_nodes(e, nodes, names, value, size, category, legend = TRUE)

e_graph_edges(e, edges, source, target)
```

**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.
nodes	Data.frame of nodes.
names	Names of nodes, unique.
value	values of nodes.
size	Size of nodes.
category	Group of nodes (i.e.: group membership).
legend	Whether to add serie to legend.
edges	Data.frame of edges.
source, target	Column names of source and target.

**See Also**

<https://ecomfe.github.io/echarts-doc/public/en/option.html#series-graph>, [e\\_modularity](#)

**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)

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

### Functions

- `g_graphic_g` to initialise graphics, entirely optional.
- `g_group_g` to create group, the children of which will share attributes.
- `g_image_g` to a png or jpg image.
- `g_text_g` to add text.
- `g_rect_g` to add a rectangle.
- `g_circle_g` to add a circle.
- `g_ring_g` to add a ring.
- `g_sector_g`
- `g_arc_g` to create an arc.
- `g_polygon_g` to create a polygon.
- `g_polyline_g` to create a polyline.
- `g_line_g` to draw a line.
- `g_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 <a href="#">e_charts</a> .
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](#).

**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	<i>Heatmap</i>
-----------	----------------

---

**Description**

Draw heatmap by coordinates.

**Usage**

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

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

**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
y, z	Coordinates and values.
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\_highlight\_p

*Highlight & Downplay Proxy*


---

### 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 <a href="#">echarts4rProxy</a> .
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(displacement) %>%
      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 = "displacement")
  })

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

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 <a href="#">e_charts</a> .
serie	Column name of serie to plot.
breaks	Passed to <a href="#">hist</a> .
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 <a href="#">e_line</a> .

**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_labels	<i>Format labels</i>
----------	----------------------

---

### Description

Format labels

### Usage

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

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
show	Set to TRUE to show the labels.
position	Position of labels, see <a href="#">official documentation</a> for the full list of options.
...	Any other options see <a href="#">documentation</a> 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	<i>Leaflet</i>
-----------	----------------

---

### 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> .
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://ecomfe.github.io/echarts-examples/",
  "public/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 <a href="#">e_charts</a> .
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	<i>Line</i>
--------	-------------

---

### 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 <a href="#">e_charts</a> .
serie	Column name of serie to plot.
bind	Binding between datasets, namely for use of <a href="#">e_brush</a> .
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	<i>Lines</i>
---------	--------------

---

### 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 <a href="#">e_charts</a> .
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 <a href="#">e_charts</a> .
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 <a href="#">e_charts</a> .
data	A list.
coord_system	Coordinate system to plot against.
...	Any other options, see this <a href="#">example</a> for possible options, as 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 <a href="#">e_charts</a> .
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**

<https://github.com/ecomfe/echarts-liquidfill>

**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 `e_charts`.

**list** A list of options passed to `setOptions`.

**append** if TRUE the list is appended to the options, otherwise it *overwrites* 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, ...)
```

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

```
e_loess(e, formula, name = NULL, legend = TRUE, symbol = "none",
         smooth = TRUE, x_index = 0, y_index = 0, ...)
```

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
formula	formula to pass to <a href="#">lm</a> .
name	name of the serie.
legend	Whether to add serie to legend.
symbol	Symbol to use in <a href="#">e_line</a> .
smooth	Whether to smooth the line.
...	Additional arguments to pass to <a href="#">e_line</a> .
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\_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_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 <code>echarts4r</code> object as returned by <code>e_charts</code> .
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](https://ecomfe.github.io/echarts-doc/public/en/option.html#series-map), <https://ecomfe.github.io/echarts-doc/public/en/option.html#series-map>, <http://echarts.baidu.com/option-gl.html#series-map3D>

### 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://ecomfe.github.io/echarts-examples/",
    "public/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 <http://geojson.org/> map.

## Usage

```
e_map_register(e, name, json)
```

**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
name	Name of map, to use in <a href="#">e_map</a> .
json	<a href="http://geojson.org/">http://geojson.org/</a> .

**Examples**

```
## Not run:
json <- jsonlite::read_json("http://www.echartsjs.com/gallery/data/asset/geo/USA.json")

USArrests %>%
  dplyr::mutate(states = row.names(.)) %>%
  e_charts(states) %>%
  e_map_register("USA", json) %>%
  e_map(Murder, map = "USA") %>%
  e_visual_map(Murder)

## End(Not run)
```

---

e\_mark\_point

*Mark point*


---

**Description**

Mark points and lines.

**Usage**

```
e_mark_point(e, serie = NULL, data = NULL, ...)
```

```
e_mark_line(e, serie = NULL, data = NULL, ...)
```

```
e_mark_area(e, serie = NULL, data = NULL, ...)
```

**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
serie	Serie or vector of series to mark on passed to <a href="#">grep</a> , defaults to all series.
data	Placement.
...	Any other option to pass, check See Also section.

**See Also**

<https://ecomfe.github.io/echarts-doc/public/en/option.html#series-line.markPoint>, <https://ecomfe.github.io/echarts-doc/public/en/option.html#series-line.markLine>

**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")))
)

```

---

e\_modularity

*Modularity*


---

**Description**

Graph modularity extension will do community detection and partian 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](#).

**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)
```

```
e_parallel_(e, ..., name = NULL, rm_x = TRUE, rm_y = TRUE)
```



**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
...	Any other option to pass, check See Also section.
name	name of the serie.
rm_x, rm_y	Whether to remove x and y axis, defaults to TRUE.

**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)
```

---

e_pictorial	<i>Pictorial</i>
-------------	------------------

---

**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 <a href="#">e_charts</a> .
serie	Column name of serie to plot.
symbol	Symbol to plot.
bind	Binding between datasets, namely for use of <a href="#">e_brush</a> .
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 <code>e_charts</code> .
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() %>%
  dplyr::mutate(model = row.names(.)) %>%
  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](#).  
**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, ...)

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

**Arguments**

**e** An echarts4r object as returned by [e\\_charts](#).  
**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.

### 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	<i>Radar axis</i>
--------------	-------------------

---

### Description

Radar axis setup and options.

### Usage

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

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
index	Index of axis to customise.
...	Any other option to pass, check See Also section.

---

e_restore	<i>Restore Toolbox</i>
-----------	------------------------

---

**Description**

Restore Toolbox.

**Usage**

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

**Arguments**

e	An echarts4r object as returned by <a href="#">e_charts</a> .
btn	A <a href="#">e_button</a> 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 <code>e_charts</code> .
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**

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

**See Also**

[Additional arguments](#)

**Examples**

```
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 <a href="#">e_charts</a> .
serie	Column name of serie to plot.
size	Column name containing size of points.
bind	Binding between datasets, namely for use of <a href="#">e_brush</a> .
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 <code>e_charts</code> .
y, z	Coordinates.
color, size	Color and Size of bubbles.
bind	Binding.
coord_system	Coordinate system to use, one of <code>geo3D</code> , <code>globe</code> , or <code>cartesian3D</code> .

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 <a href="#">e_charts</a> .
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
        })
      }

      shinyApp(ui, server)

## End(Not run)

```

---

e_show_loading	<i>Loading</i>
----------------	----------------

---

## Description

Show or hide loading.

## Usage

```

e_show_loading(e, hide_overlay = TRUE, text = "loading",
  color = "#c23531", text_color = "#0000",
  mask_color = "rgba(255, 255, 255, 0.8)", zlevel = 0)

e_hide_loading(e)

```

## Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
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)
  })
}

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()
  })
}

shinyApp(ui, server)

```

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

---

e_single_axis	<i>Single Axis</i>
---------------	--------------------

---

### Description

Setup single axis.

### Usage

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

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
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 <a href="#">e_charts</a> .
serie	Column name of serie to plot.
bind	Binding between datasets, namely for use of <a href="#">e_brush</a> .
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 %>%
  dplyr::mutate(State = row.names(.)) %>%
  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	<i>Sunburst</i>
------------	-----------------

---

## Description

Build a sunburst.

## Usage

```
e_sunburst(e, parent, child, value, itemStyle, rm_x = TRUE,
           rm_y = TRUE, ...)
```

```
e_sunburst_(e, parent, child, value, itemStyle = NULL, rm_x = TRUE,
            rm_y = TRUE, ...)
```

## Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
parent, child	Edges.
value	Name of column containing values.
itemStyle	Name of column containing styles to pass to child, expects a data.frame or a list, see details.
rm_x, rm_y	Whether to remove x and y axis, defaults to TRUE.
...	Any other option to pass, check See Also section.

## Details

The `itemStyle` argument essentially is a nested data.frame with column names such as `color`, or `borderColor` as specified in the [official documentation](#).

## See Also

[Additional arguments](#)

**Examples**

```
df <- data.frame(
  parent = c("earth", "earth", "earth", "mars", "mars"),
  child = c("forest", "ocean", "iceberg", "elon", "curiosity"),
  value = ceiling(rnorm(5, 10, 2))
)

df %>%
  e_charts() %>%
  e_sunburst(parent, child, value) %>%
  e_theme("westeros")

# with itemStyle
colors <- c("red", "black", "blue")

df$color <- sample(colors, 5, replace = TRUE)
df$borderColor <- sample(colors, 5, replace = TRUE)

df %>%
  tidyr::nest(color, borderColor, .key = "style") %>% # nest
  e_charts() %>%
  e_sunburst(parent, child, value, style)
```

---

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 <a href="#">e_charts</a> .
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`.  
`...` 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 theme.

### Usage

```
e_theme(e, theme)
```

```
e_theme_custom(e, theme)
```

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
theme	Theme, see below.

### Themes

- default
- dark
- vintage
- westeros
- essos
- wonderland
- walden
- chalk
- infographic
- macarons
- roma
- shine
- purple-passion
- halloween
- auritus

### 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	<i>Title</i>
---------	--------------

---

**Description**

Add title.

**Usage**

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

**Arguments**

**e** An echarts4r object as returned by [e\\_charts](#).

**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      *Toolbox*

---

### Description

Add toolbox interface.

### Usage

```
e_toolbox_feature(e, feature, ...)
```

```
e_toolbox(e, ...)
```

### Arguments

e	An echarts4r object as returned by <a href="#">e_charts</a> .
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 %>%
  dplyr::mutate(model = row.names(.)) %>%
  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_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 <a href="#">e_charts</a> .
trigger	What triggers the tooltip, one of <code>item</code> or <code>axis</code> .
formatter	Item and Pointer formatter as returned by <a href="#">e_tooltip_item_formatter</a> , <a href="#">e_tooltip_pointer_formatter</a> , <a href="#">e_tooltip_pie_formatter</a> .
...	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, parent, child, rm_x = TRUE, rm_y = TRUE, ...)
```

```
e_tree_(e, parent, child, rm_x = TRUE, rm_y = TRUE, ...)
```

**Arguments**

e An echarts4r object as returned by [e\\_charts](#).  
 parent, child Edges.  
 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**

```
df <- data.frame(
  parent = c("earth", "earth", "forest", "forest", "ocean", "ocean", "ocean", "ocean"),
  child = c("ocean", "forest", "tree", "sasquatch", "fish", "seaweed", "mantis shrimp", "sea monster")
)

df %>%
  e_charts() %>%
  e_tree(parent, child)
```

---

e\_treemap

*Treemap*


---

**Description**

Build a treemap.

**Usage**

```
e_treemap(e, parent, child, value, rm_x = TRUE, rm_y = TRUE, ...)
```

```
e_treemap_(e, parent, child, value, rm_x = TRUE, rm_y = TRUE, ...)
```

**Arguments**

e An echarts4r object as returned by [e\\_charts](#).  
 parent, child Edges.  
 value Value of edges.  
 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**

```
df <- data.frame(
  parent = c("earth", "earth", "earth", "mars", "mars"),
  child = c("forest", "ocean", "iceberg", "elon", "curiosity"),
  value = ceiling(rnorm(5, 10, 2))
)

df %>%
  e_charts() %>%
  e_treemap(parent, child, value)
```

---

e_utc	<i>Use UTC</i>
-------	----------------

---

**Description**

Use UTC

**Usage**

```
e_utc(e)
```

**Arguments**

e                    An echarts4r object as returned by [e\\_charts](#).

---

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 <a href="#">e_charts</a> .
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 <code>e_charts</code> .
...	Any options, see <a href="#">official documentation</a>
btn	A <code>e_button</code> 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](#).  
**...** 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

*Nodes Adjacency*


---

### 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`.  
`...` 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

*Highlight & Downplay*


---

**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`.

`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, renderer = "canvas", timeline = FALSE, ...)
```

```
e_charts_(data, x = NULL, width = NULL, height = NULL,
  elementId = NULL, dispose = TRUE, renderer = "canvas",
  timeline = FALSE, ...)
```

```
e_chart(data, x, width = NULL, height = NULL, elementId = NULL,
  dispose = TRUE, renderer = "canvas", timeline = FALSE, ...)
```

```
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.

renderer	Renderer, takes canvas (default) or svg.
timeline	Set to TRUE to build a timeline, see timeline section.
...	Any other argument.
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	<i>Legend</i>
---------------	---------------

---

**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 <code>echarts4r</code> object as returned by <code>e_charts</code> .
<code>name</code>	Legend name.
<code>btn</code>	A <code>e_button</code> id.
<code>scroll_index</code>	Controlle the scrolling of legend when <code>type = "scroll"</code> in <code>e_legend</code> .
<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

e	An echarts4r object as returned by <a href="#">e_charts</a> .
token	Your mapbox token from <a href="#">mapbox</a> .
...	Any option.

### Note

Mapbox may not work properly in the RSudio console.

### See Also

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

### Examples

```
## Not run:
url <- paste0("https://ecomfe.github.io/echarts-examples/",
             "public/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 <a href="#">e_charts</a> .
...	Any options, see <a href="#">official documentation</a>
btn	A <a href="#">e_button</a> 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")
```

---

pie_action	<i>Select &amp; 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 <a href="#">e_charts</a> .
...	Any options, see <a href="#">official documentation</a>
btn	A <a href="#">e_button</a> id.

### Examples

```
mtcars %>%
  head() %>%
  dplyr::mutate(model = row.names(.)) %>%
  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](#).
- 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")
```

timeline-opts      *Timeline*

**Description**

Set timeline options

**Usage**

```
e_timeline_opts(e, axis_type = "category", ...)

e_timeline_serie(e, ...)
```

**Arguments**

- e An echarts4r object as returned by [e\\_charts](#).
- axis\_type Type of axis, time, value, or category
- ... Named options.

**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	<i>Show &amp; Hide Tooltip</i>
----------------	--------------------------------

---

**Description**

Show or hide tooltip.

**Usage**

```
e_showtip(e, ..., btn = NULL)
```

```
e_hidetip(e, ..., btn = NULL)
```

**Arguments**

e	An echarts4r object as returned by <code>e_charts</code> .
...	Any options, see <a href="#">official documentation</a>
btn	A <code>e_button</code> 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

\*Topic **interval**

  e\_facet, 34

angle\_axis, 4

band, 5

browsable, 7

callbacks, 6

colorRampPalette, 28

connections, 6

corrMatOrder, 29

countrycode, 30

div, 7

e\_add, 9

e\_angle\_axis (angle\_axis), 4

e\_angle\_axis\_ (angle\_axis), 4

e\_animation, 10

e\_append1\_p, 9, 11

e\_append1\_p\_ (e\_append1\_p), 11

e\_append2\_p, 9

e\_append2\_p (e\_append1\_p), 11

e\_append2\_p\_ (e\_append1\_p), 11

e\_arc\_g (e\_graphic\_g), 47

e\_area, 13, 108

e\_area\_ (e\_area), 13

e\_aria, 14

e\_arrange (connections), 6

e\_axis, 15, 39

e\_axis\_3d, 16

e\_axis\_formatter, 15

e\_axis\_formatter (e\_axis), 15

e\_axis\_pointer, 17

e\_band (band), 5

e\_band\_ (band), 5

e\_bar, 18, 34, 108

e\_bar\_ (e\_bar), 18

e\_bar\_3d, 19, 108

e\_bar\_3d\_ (e\_bar\_3d), 19

e\_bezier\_curve\_g (e\_graphic\_g), 47

e\_boxplot, 21, 108

e\_boxplot\_ (e\_boxplot), 21

e\_brush, 13, 18, 22, 25, 59, 73, 82, 92

e\_button, 23, 79, 104–107, 109, 111, 112, 114

e\_calendar, 23

e\_candle, 24, 108

e\_candle\_ (e\_candle), 24

e\_capture, 25

e\_chart (init), 107

e\_charts, 4–7, 9, 11, 13–15, 17–19, 21–27,  
29, 31, 33–36, 39–45, 48–51, 55–60,  
62, 64–67, 70, 71, 73, 76–82, 84, 86,  
89, 91–99, 101–107, 109–114

e\_charts (init), 107

e\_charts\_ (init), 107

e\_circle\_g (e\_graphic\_g), 47

e\_cloud, 26

e\_cloud\_ (e\_cloud), 26

e\_color, 27

e\_color\_range, 28

e\_color\_range\_ (e\_color\_range), 28

e\_common, 29

e\_connect (connections), 6

e\_connect\_group (connections), 6

e\_correlations, 29

e\_country\_names, 30, 43, 45, 68

e\_country\_names\_ (e\_country\_names), 30

e\_data (init), 107

e\_datazoom, 30

e\_density, 108

e\_density (e\_histogram), 54

e\_density\_ (e\_histogram), 54

e\_disconnect\_group (connections), 6

e\_dispatch\_action\_p, 9, 31

e\_downplay (highlight\_action), 106

e\_downplay\_p, 9

e\_downplay\_p (e\_highlight\_p), 53

e\_draft, 32

- e\_effect\_scatter, [108](#)
- e\_effect\_scatter (e\_scatter), [81](#)
- e\_effect\_scatter\_ (e\_scatter), [81](#)
- e\_error\_bar, [33](#)
- e\_error\_bar\_ (e\_error\_bar), [33](#)
- e\_facet, [34](#)
- e\_flip\_coords, [35](#)
- e\_flow\_gl, [35](#)
- e\_flow\_gl\_ (e\_flow\_gl), [35](#)
- e\_focus\_adjacency, [9](#)
- e\_focus\_adjacency (graph\_action), [105](#)
- e\_focus\_adjacency\_p, [37](#)
- e\_format\_axis, [39](#)
- e\_format\_x\_axis (e\_format\_axis), [39](#)
- e\_format\_y\_axis (e\_format\_axis), [39](#)
- e\_funnel, [9](#), [40](#)
- e\_funnel\_ (e\_funnel), [40](#)
- e\_gauge, [41](#), [108](#)
- e\_gauge\_ (e\_gauge), [41](#)
- e\_geo, [42](#)
- e\_geo\_3d, [43](#)
- e\_geo\_3d\_ (e\_geo\_3d), [43](#)
- e\_get\_data, [44](#)
- e\_glm, [108](#)
- e\_glm (e\_lm), [66](#)
- e\_globe, [44](#)
- e\_graph, [45](#), [105](#)
- e\_graph\_edges (e\_graph), [45](#)
- e\_graph\_gl (e\_graph), [45](#)
- e\_graph\_nodes (e\_graph), [45](#)
- e\_graphic\_g, [47](#)
- e\_grid, [49](#)
- e\_grid\_3d, [50](#)
- e\_group (connections), [6](#)
- e\_group\_g (e\_graphic\_g), [47](#)
- e\_heatmap, [29](#), [51](#), [108](#)
- e\_heatmap\_ (e\_heatmap), [51](#)
- e\_hide\_loading (e\_show\_loading), [89](#)
- e\_hidetip (tooltip\_action), [114](#)
- e\_hidetip\_p, [9](#)
- e\_hidetip\_p (e\_showtip\_p), [87](#)
- e\_highlight (highlight\_action), [106](#)
- e\_highlight\_p, [9](#), [53](#)
- e\_histogram, [54](#), [108](#)
- e\_histogram\_ (e\_histogram), [54](#)
- e\_image\_g (e\_graphic\_g), [47](#)
- e\_labels, [56](#)
- e\_leaflet, [56](#)
- e\_leaflet\_tile (e\_leaflet), [56](#)
- e\_legend, [57](#)
- e\_legend\_scroll (legend\_action), [109](#)
- e\_legend\_select (legend\_action), [109](#)
- e\_legend\_toggle\_select (legend\_action), [109](#)
- e\_legend\_unselect (legend\_action), [109](#)
- e\_line, [12](#), [55](#), [59](#), [66](#), [108](#)
- e\_line\_ (e\_line), [59](#)
- e\_line\_3d, [12](#), [108](#)
- e\_line\_3d (e\_lines\_3d), [61](#)
- e\_line\_3d\_ (e\_lines\_3d), [61](#)
- e\_line\_g (e\_graphic\_g), [47](#)
- e\_lines, [60](#), [108](#)
- e\_lines\_ (e\_lines), [60](#)
- e\_lines\_3d, [61](#), [108](#)
- e\_lines\_3d\_ (e\_lines\_3d), [61](#)
- e\_lines\_gl, [64](#)
- e\_liquid, [64](#)
- e\_liquid\_ (e\_liquid), [64](#)
- e\_list, [65](#)
- e\_lm, [66](#), [108](#)
- e\_loess, [108](#)
- e\_loess (e\_lm), [66](#)
- e\_map, [67](#), [70](#), [108](#)
- e\_map\_ (e\_map), [67](#)
- e\_map\_3d, [108](#)
- e\_map\_3d (e\_map), [67](#)
- e\_map\_3d\_ (e\_map), [67](#)
- e\_map\_3d\_custom (e\_map), [67](#)
- e\_map\_register, [69](#), [111](#)
- e\_map\_select (map\_actions), [111](#)
- e\_map\_toggle\_select (map\_actions), [111](#)
- e\_map\_unselect (map\_actions), [111](#)
- e\_mapbox (mapbox), [110](#)
- e\_mark\_area (e\_mark\_point), [70](#)
- e\_mark\_line (e\_mark\_point), [70](#)
- e\_mark\_point, [70](#)
- e\_modularity, [46](#), [71](#)
- e\_off (callbacks), [6](#)
- e\_on (callbacks), [6](#)
- e\_parallel, [72](#)
- e\_parallel\_ (e\_parallel), [72](#)
- e\_pictorial, [73](#), [108](#)
- e\_pictorial\_ (e\_pictorial), [73](#)
- e\_pie, [75](#), [99](#), [108](#), [112](#)
- e\_pie\_ (e\_pie), [75](#)
- e\_pie\_select (pie\_action), [112](#)

- e\_pie\_unselect (pie\_action), 112
- e\_polar, 76
- e\_polygon\_g (e\_graphic\_g), 47
- e\_polyline\_g (e\_graphic\_g), 47
- e\_radar, 77
- e\_radar\_ (e\_radar), 77
- e\_radar\_opts, 78
- e\_radius\_axis (radius\_axis), 112
- e\_radius\_axis\_ (radius\_axis), 112
- e\_rect\_g (e\_graphic\_g), 47
- e\_restore, 79
- e\_ring\_g (e\_graphic\_g), 47
- e\_river, 79
- e\_river\_ (e\_river), 79
- e\_rm\_axis (e\_axis), 15
- e\_sankey, 80
- e\_sankey\_ (e\_sankey), 80
- e\_scale (e\_scatter), 81
- e\_scatter, 12, 81, 108
- e\_scatter\_ (e\_scatter), 81
- e\_scatter\_3d, 12, 84, 108
- e\_scatter\_3d\_ (e\_scatter\_3d), 84
- e\_scatter\_gl, 86, 108
- e\_scatter\_gl\_ (e\_scatter\_gl), 86
- e\_sector\_g (e\_graphic\_g), 47
- e\_show\_loading, 89
- e\_showtip (tooltip\_action), 114
- e\_showtip\_p, 9, 87
- e\_single\_axis, 91
- e\_step, 92, 108
- e\_step\_ (e\_step), 92
- e\_sunburst, 93
- e\_sunburst\_ (e\_sunburst), 93
- e\_surface, 94
- e\_surface\_ (e\_surface), 94
- e\_text\_g (e\_graphic\_g), 47
- e\_text\_style, 95
- e\_theme, 27, 96
- e\_theme\_custom (e\_theme), 96
- e\_timeline\_opts (timeline-opts), 113
- e\_timeline\_serie (timeline-opts), 113
- e\_title, 97
- e\_toolbox (e\_toolbox\_feature), 98
- e\_toolbox\_feature, 31, 98
- e\_tooltip, 99, 114
- e\_tooltip\_item\_formatter, 99
- e\_tooltip\_item\_formatter (e\_tooltip), 99
- e\_tooltip\_pie\_formatter, 99
- e\_tooltip\_pie\_formatter (e\_tooltip), 99
- e\_tooltip\_pointer\_formatter, 99
- e\_tooltip\_pointer\_formatter (e\_tooltip), 99
- e\_tree, 100
- e\_tree\_ (e\_tree), 100
- e\_treemap, 101
- e\_treemap\_ (e\_treemap), 101
- e\_unfocus\_adjacency, 9
- e\_unfocus\_adjacency (graph\_action), 105
- e\_unfocus\_adjacency\_p (e\_focus\_adjacency\_p), 37
- e\_utc, 102
- e\_visual\_map, 29, 102
- e\_visual\_map\_ (e\_visual\_map), 102
- e\_visual\_map\_range, 104
- e\_x\_axis (e\_axis), 15
- e\_x\_axis\_3d (e\_axis\_3d), 16
- e\_y\_axis (e\_axis), 15
- e\_y\_axis\_3d (e\_axis\_3d), 16
- e\_z\_axis (e\_axis), 15
- e\_z\_axis\_3d (e\_axis\_3d), 16
- e\_zoom, 105
- echarts4r-shiny, 8
- echarts4rOutput (echarts4r-shiny), 8
- echarts4rProxy, 12, 31, 37, 53, 87
- echarts4rProxy (echarts4r-shiny), 8
- graph\_action, 105
- grep, 70
- highlight\_action, 106
- hist, 55
- init, 107
- JS, 6
- legend\_action, 109
- lm, 66
- map\_actions, 111
- mapbox, 110
- pie\_action, 112
- radius\_axis, 112
- renderEcharts4r (echarts4r-shiny), 8
- tags, 23
- timeline-opts, 113
- tooltip\_action, 114