Package: rdeck (via r-universe)

August 14, 2024

Type Package

Title Deck.gl Widget

Version 0.5.2.9000

Description Deck.gl widget for R.

License MIT + file LICENSE

URL https://qfes.github.io/rdeck, https://github.com/qfes/rdeck

BugReports https://github.com/qfes/rdeck/issues

Depends R (>= 3.3.0)

Imports dplyr, generics, htmlwidgets (>= 1.5.1), jsonlite, lifecycle, magrittr, purrr (>= 1.0), R6, rlang (>= 1.0), scales, sf (>= 0.9.3), stylebox (>= 0.2.0), tidyassert (>= 0.3.0), tidyselect, uuid, vctrs (>= 0.6), wk (>= 0.9.2), yyjsonr

Suggests geojsonsf, knitr, png, RcppSimdJson, rmarkdown, roxyglobals, shiny, testthat (>= 3.0.0), urltools, viridis

VignetteBuilder knitr

Remotes anthonynorth/stylebox, qfes/tidyassert

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c(``collate", ``namespace", ``rd", ``roxyglobals::global_roclet"))

RoxygenNote 7.3.2

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Repository https://anthonynorth.r-universe.dev

RemoteUrl https://github.com/qfes/rdeck

RemoteRef HEAD

RemoteSha 9773f16aab1c4d176dbf17f576e406c3285f5c26

Contents

`arc_layer` *Arc Layer*

Description

Arc Layer

Usage

```
add_arc_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "ArcLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_source_position = source_position,
```

```
  get_target_position = target_position,
  get_source_color = "#000000ff",
  get_target_color = "#000000ff",
  get_width = 1,
  get_height = 1,
  get_tilt = 0,
  great_circle = FALSE,
  width_units = "pixels",
  width_scale = 1,
  width_min_pixels = 0,
  width_max_pixels = 9007199254740991,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_arc_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_source_position = cur_value(),
  get_target_position = cur_value(),
  get_source_color = cur_value(),
  get_target_color = cur_value(),
  get_width = cur_value(),
  get_height = cur_value(),
  get_tilt = cur_value(),
  great_circle = cur_value(),
  width_units = cur_value(),
  width_scale = cur_value(),
  width_min_pixels = cur_value(),
  width_max_pixels = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
get_source_position	<accessor> The feature source positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.
get_target_position	<accessor> The feature target positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.
get_source_color	<accessor | scale | color> The colour of the *source end* of the arc. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_target_color
: <accessor | scale | color> The colour of the *target end* of the arc. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_width
: <accessor | scale | number> The width of each object, in units specified by `width_scale`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_height
: <accessor | scale | number> The multiplier of layer of layer height. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers. A value of 0 will make the layer flat.

get_tilt
: <accessor | number> Tilts the arcs by the specified number of degrees (between `c(-90, 90)`). Accepts a single numeric value or a tidy-eval column of numbers.

great_circle
: <boolean> If `TRUE`, create the arc along the shortest path on the earth surface.

width_units
: <"pixels" | "common" | "meters"> The units of the `line_width`.

width_scale
: <number> The scaling multiplier for the width of each line.

width_min_pixels
: <number> The minimum line width in pixels.

width_max_pixels
: <number> The maximum line width in pixels.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:
  - `normal`: Normal blending doesn't alter colours of overlapping objects.
  - `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if `pickable == TRUE`. Supports tidy-select if a `data` is a `data.frame`. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/arc-layer.md

Other core-layers: `bitmap_layer`, `column_layer`, `geojson_layer`, `grid_cell_layer`, `icon_layer`, `line_layer`, `path_layer`, `point_cloud_layer`, `polygon_layer`, `scatterplot_layer`, `solid_polygon_layer`, `text_layer`

Other layers: `bitmap_layer`, `column_layer`, `contour_layer`, `cpu_grid_layer`, `geojson_layer`, `gpu_grid_layer`, `great_circle_layer`, `grid_cell_layer`, `grid_layer`, `h3_cluster_layer`, `h3_hexagon_layer`, `heatmap_layer`, `hexagon_layer`, `icon_layer`, `line_layer`, `mvt_layer`, `path_layer`, `point_cloud_layer`, `polygon_layer`, `quadkey_layer`, `s2_layer`, `scatterplot_layer`, `scenegraph_layer`, `screen_grid_layer`, `simple_mesh_layer`, `solid_polygon_layer`, `terrain_layer`, `text_layer`, `tile_3d_layer`, `tile_layer`, `trips_layer`

bitmap_layer *Bitmap Layer*

Description

Bitmap Layer

Usage

```
add_bitmap_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "BitmapLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  image = NULL,
  bounds = c(1, 0, 0, 1),
  desaturate = 0,
  transparent_color = "#00000000",
  tint_color = "#ffffff",
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_bitmap_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
```

```
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  image = cur_value(),
  bounds = cur_value(),
  desaturate = cur_value(),
  transparent_color = cur_value(),
  tint_color = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

image <string | array> The image to display. Either a string interpreted as a URL or Data URL, or an image bitmap.

bounds <rct/st_bbox/wk-geometry> The bounds of the image to fit x,y coordinates into. Requires CRS EPSG:4326.

desaturate <number> The desaturation of the bitmap. Between 0 and 1, being the original colour, 1 being greyscale.

transparent_color <color> The colour to use for transparent pixels.

tint_color <color> The colour to tint the bitmap by. Alpha channel is ignored if supplied.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle <boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/bitmap-layer.md

Other core-layers: arc_layer, column_layer, geojson_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

breaks_linear *Breaks linear*

Description

Generate a breaks vector of size n with linearly spaced breaks.

Usage

```
breaks_linear(n = 10)
```

Arguments

n	<int> the size of the output vector. Ouput size will be at least length-2 for finite input.

Details

If input range is non-finite, an empty vector is returned.

See Also

Other breaks: breaks_log(), breaks_manual(), breaks_power(), breaks_symlog(), breaks_trans()

Examples

```
breaks_linear(5)(-10:10)
breaks_linear()(-1:1)
```

---

breaks_log *Breaks log*

Description

Generate a breaks vector of size n with log log_trans spaced breaks.

Usage

```
breaks_log(n = 10, base = exp(1))
```

Arguments

n	<int> the size of the output vector. Ouput size will be at least length-2 for finite input.
base	<number> The log base

Details

Input range must be finite and either be strictly negative or strictly positive; it must not cross, nor include 0.

If input range is non-finite, an empty vector is returned.

Note

This function has a different goal to scales::breaks_log: it produces evenly spaced log-breaks for use with d3-scale, it doesn't produce breaks with pretty labels.

See Also

Other breaks: `breaks_linear()`, `breaks_manual()`, `breaks_power()`, `breaks_symlog()`, `breaks_trans()`

Examples

```
breaks_log(5)(-10:-1)
breaks_log(5)(1:10)
```

`breaks_manual` *Breaks manual*

Description

Generate a breaks vector with pre-determined threshold values.

Usage

```
breaks_manual(thresholds = 0.5)
```

Arguments

`thresholds`	<numeric> a numeric vector of *ordered*, finite thresholds.

Details

Wraps `thresholds` in the range of input data (which may be null or non-finite). Thresholds that fall outside the range of input are omitted from the output.

See Also

Other breaks: `breaks_linear()`, `breaks_log()`, `breaks_power()`, `breaks_symlog()`, `breaks_trans()`

Examples

```
breaks_manual(0)(-10:10)
breaks_manual(c(-10, 0, 10))(-10:10)
breaks_manual(-1:1 * 1e6)(NULL)
```

breaks_power *Breaks power*

Description

Generate a breaks vector of size n with exponentially power_trans spaced breaks.

Usage

```
breaks_power(n = 10, exponent = 0.5)
```

Arguments

n	<int> the size of the output vector. Ouput size will be at least length-2 for finite input.
exponent	<number> The power exponent

Details

If input range is non-finite, an empty vector is returned.

See Also

Other breaks: breaks_linear(), breaks_log(), breaks_manual(), breaks_symlog(), breaks_trans()

Examples

```
breaks_power(5)(-10:10)
breaks_power(5, exponent = 1 / 3)(-1:1)
```

breaks_symlog *Breaks symlog*

Description

Generate a breaks vector of size n with log1p symlog_trans spaced breaks.

Usage

```
breaks_symlog(n = 10)
```

Arguments

n	<int> the size of the output vector. Ouput size will be at least length-2 for finite input.

Details

If input range is non-finite, an empty vector is returned.

See Also

Other breaks: `breaks_linear()`, `breaks_log()`, `breaks_manual()`, `breaks_power()`, `breaks_trans()`

Examples

```
breaks_symlog(5)(-10:10)
breaks_symlog(5)(0:10)
```

`breaks_trans` *Breaks trans*

Description

Generate a breaks vector of size `n` with evenly spaced breaks in `trans` domain.

Usage

```
breaks_trans(n = 10, trans)
```

Arguments

`n`	<int> the size of the output vector. Ouput size will be at least length-2 for finite input.
`trans`	<scales::trans> an *invertible* transformer.

Details

Breaks are generated by transforming the input range to `trans` domain, generating a regular sequence of size `n` for the transformed range, then inverting the transform.

The input range must be finite for both the input domain *and* the `trans` domain. If input is not finite, an empty vector is returned.

See Also

Other breaks: `breaks_linear()`, `breaks_log()`, `breaks_manual()`, `breaks_power()`, `breaks_symlog()`

Examples

```
breaks_trans(trans = scales::identity_trans())(-10:10)
breaks_trans(trans = symlog_trans())(-10:10)
breaks_trans(trans = power_trans())(-10:10)
breaks_manual(c(-10, 0, 10))(-10:10)
breaks_manual(-1:1 * 1e6)(NULL)
```

column_layer *Column Layer*

Description

Column Layer

Usage

```
add_column_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "ColumnLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  disk_resolution = 20,
  vertices = NULL,
  radius = 1000,
  angle = 0,
  offset = c(0, 0),
  coverage = 1,
  elevation_scale = 1,
  radius_units = "meters",
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  extruded = FALSE,
  wireframe = FALSE,
  filled = TRUE,
  stroked = FALSE,
  get_position = position,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  blending_mode = "normal",
```

```
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_column_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  disk_resolution = cur_value(),
  vertices = cur_value(),
  radius = cur_value(),
  angle = cur_value(),
  offset = cur_value(),
  coverage = cur_value(),
  elevation_scale = cur_value(),
  radius_units = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  extruded = cur_value(),
  wireframe = cur_value(),
  filled = cur_value(),
  stroked = cur_value(),
  get_position = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.

`...` Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be `camelCased` when serialised. A warning is raised when dots are used, warning class `rdeck_dots_nonempty`.

`id` <string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to `uuid::UUIDgenerate()`, but should be explicitly defined for updatable layers in a shiny application.

`name` <string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

`group_name` <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if `visibility_toggle = TRUE`.

`data` <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

`visible` <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

`pickable` <boolean> Determines if the layer responds to pointer / touch events.

`opacity` <number> Determines the layer's opacity.

`wrap_longitude` <boolean> Normalises geometry longitudes.

`position_format`
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`
<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`disk_resolution`
<number> The number of sides to render the disk as. The disk is a regular polygon that fits inside the given `radius`. A higher resolution will yield a smoother look close-up, but also need more resources to render.

`vertices` `matrix` Replace the default geometry (regular polygon that fits inside the unit circle) with a custom one. The length of the array must be at least `disk_resolution`. Each vertex is a row `c(x, y)` that is the offset from the instance position, relative to the radius.

`radius` <number> The radius of the column in metres.

`angle` <number> The disk rotation, counter-clockwise in radians.

`offset` <number> The disk offset from the position, relative to the radius.

`coverage` <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by `coverage * radius`.

elevation_scale
: <number> The elevation multiplier.

radius_units
: <"pixels" | "common" | "meters"> The units of point radius.

line_width_units
: <"pixels" | "common" | "meters"> The units of outline width. Applied when extruded == FALSE and stroked == TRUE.

line_width_scale
: <number> The line width multiplier.

line_width_min_pixels
: <number> The minimum line width in pixels.

line_width_max_pixels
: <number> The maximum line width in pixels.

extruded
: <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

wireframe
: <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

filled
: <boolean> If TRUE, draw the filled area of each point.

stroked
: <boolean> If TRUE, draw an outline around each object.

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_fill_color
: <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_color
: <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_width
: <accessor | scale | number> The outline of the object in units specified by line_width_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_elevation
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

material
: <boolean>

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/column-layer.md

Other core-layers: arc_layer, bitmap_layer, geojson_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

contour_layer *Contour Layer*

Description

Contour Layer

Usage

```
add_contour_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "ContourLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  cell_size = 1000,
  get_position = position,
  get_weight = 1,
  gpu_aggregation = TRUE,
  aggregation = "SUM",
  contours = c(list(threshold = 1)),
  z_offset = 0.005,
  blending_mode = "normal",
  visibility_toggle = TRUE
```

```
)

update_contour_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  cell_size = cur_value(),
  get_position = cur_value(),
  get_weight = cur_value(),
  gpu_aggregation = cur_value(),
  aggregation = cur_value(),
  contours = cur_value(),
  z_offset = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

cell_size <number> The size of each cell in metres.

get_position <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_weight <accessor | scale | number> The weight of each object. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

gpu_aggregation
<boolean> If TRUE, aggregation is performed on GPU if supported. Requires WebGL2.

aggregation <"SUM" | "MEAN" | "MIN" | "MAX"> Defines the aggregation function.

contours array

z_offset <number> A very small z offset that is added for each vertex of a contour (isoline or isoband). Needed to control the layout of the contours. See deck.gl contours

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/contour-layer.md

Other aggregation-layers: cpu_grid_layer, gpu_grid_layer, grid_layer, heatmap_layer, hexagon_layer, screen_grid_layer

Other layers: arc_layer, bitmap_layer, column_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

`cpu_grid_layer` *CPU Grid Layer*

Description

CPU Grid Layer

Usage

```
add_cpu_grid_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "CPUGridLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  color_domain = NULL,
 color_range = c("#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"),
  get_color_value = NULL,
  get_color_weight = 1,
  color_aggregation = "SUM",
  lower_percentile = 0,
  upper_percentile = 100,
  color_scale_type = "quantize",
  elevation_domain = NULL,
  elevation_range = c(0, 1000),
  get_elevation_value = NULL,
  get_elevation_weight = 1,
  elevation_aggregation = "SUM",
  elevation_lower_percentile = 0,
  elevation_upper_percentile = 100,
  elevation_scale = 1,
```

```
  elevation_scale_type = "linear",
  cell_size = 1000,
  coverage = 1,
  get_position = position,
  extruded = FALSE,
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE
)

update_cpu_grid_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  color_domain = cur_value(),
  color_range = cur_value(),
  get_color_value = cur_value(),
  get_color_weight = cur_value(),
  color_aggregation = cur_value(),
  lower_percentile = cur_value(),
  upper_percentile = cur_value(),
  color_scale_type = cur_value(),
  elevation_domain = cur_value(),
  elevation_range = cur_value(),
  get_elevation_value = cur_value(),
  get_elevation_weight = cur_value(),
  elevation_aggregation = cur_value(),
  elevation_lower_percentile = cur_value(),
  elevation_upper_percentile = cur_value(),
  elevation_scale = cur_value(),
  elevation_scale_type = cur_value(),
  cell_size = cur_value(),
  coverage = cur_value(),
  get_position = cur_value(),
  extruded = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
```

```
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
color_domain	<number> The colour scale domain, default is set to the range of aggregated weights in each bin.
color_range	<color> The colour palette. color_domain is divided into length(color_range) equal segments, each mapped to one color in color_range.
get_color_value	<JS> After data objects are aggregated into bins, this accessor is called on each bin to get the value that its colour is based on. If supplied, this will override the effect of get_color_weight and color_aggregation props. See deck.gl docs for details.

get_color_weight
: <accessor | scale | number> The weight of each object used to calculate the colour value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

color_aggregation
: <"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's colour.

lower_percentile
: <number> Between 0 and 100. Filter bins and re-calculate colour by lower_percentile. Cells with value < lower_percentile will be hidden.

upper_percentile
: <number> Between 0 and 100. Filter bins and re-calculate colour by upper_percentile. Cells with value < upper_percentile will be hidden.

color_scale_type
: <"quantize" | "linear" | "quantile" | "ordinal"> The scaling function used to determine the colour of a the grid cell.

elevation_domain
: <number> The elevation scale input domain. Defaults to the range of the aggregated weights in each bin.

elevation_range
: <number> The elevation scale output range.

get_elevation_value
: <JS> After data objects are aggregated into bins, this accessor is called on each bin to get the value that its elevation is based on. If supplied, this will override the effect of get_elevation_weight and elevation_aggregation props. See deck.gl docs for details.

get_elevation_weight
: <accessor | scale | 'numeric> The weight of each object used to calculate the elevation value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

elevation_aggregation
: <"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's elevation value.

elevation_lower_percentile
: <number> Between 0 and 100. Filter bins and re-calculate elevation by elevation_lower_percentile. Cells with value < elevation_lower_percentile will be hidden.

elevation_upper_percentile
: <number> Between 0 and 100. Filter bins and re-calculate elevation by elevation_upper_percentile. Cells with value < elevation_upper_percentile will be hidden.

elevation_scale
: <number> The elevation multiplier.

elevation_scale_type
: <"quantize" | "linear" | "quantile" | "ordinal"> The scaling function used to determine the elevation of a the grid cell.

cell_size
: <number> The size of each cell in metres.

coverage
: <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by coverage * radius.

get_position <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

extruded <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

material <boolean>

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle <boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/cpu-grid-layer.md

Other aggregation-layers: contour_layer, gpu_grid_layer, grid_layer, heatmap_layer, hexagon_layer, screen_grid_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

editor_options *Polygon editor options*

Description

Options for the polygon editor

Usage

```
editor_options(mode = cur_value(), features = cur_value())
```

Arguments

mode <editor-mode> The polygon editor mode. One of:

- view: editor is in readonly mode
- select: select/unselect features

- `modify`: add/move/delete vertices
- `transform`: move/scale/rotate selected features
- `point`: draw points
- `linestring`: draw linestrings by clicking each vertex
- `polygon`: draw polygons by clicking each vertex
- `lasso`: freehand polygon draw by click-dragging

`features` <wk-geometry> Features with which to initialise the editor. Requires CRS EPSG:4326.

`format_number` *Format number*

Description

Format numeric and integer values.

Usage

```
format_number(tick, digits = 2)
```

Arguments

`tick`	a vector of tick values
`digits`	the number of digits to output

`geojson_layer` *GeoJson Layer*

Description

GeoJson Layer

Usage

```
add_geojson_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "GeoJsonLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
```

```
    wrap_longitude = FALSE,
    position_format = "XY",
    color_format = "RGBA",
    auto_highlight = FALSE,
    highlight_color = "#00008080",
    filled = TRUE,
    stroked = TRUE,
    line_width_max_pixels = 9007199254740991,
    line_width_min_pixels = 0,
    line_width_scale = 1,
    line_width_units = "meters",
    point_radius_max_pixels = 9007199254740991,
    point_radius_min_pixels = 0,
    point_radius_scale = 1,
    point_radius_units = "meters",
    point_antialiasing = TRUE,
    point_billboard = FALSE,
    get_fill_color = "#000000ff",
    get_line_color = "#000000ff",
    get_line_width = 1,
    get_point_radius = 1,
    icon_atlas = NULL,
    icon_mapping = list(),
    icon_size_max_pixels = 9007199254740991,
    icon_size_min_pixels = 0,
    icon_size_scale = 1,
    icon_size_units = "pixels",
    icon_alpha_cutoff = 0.05,
    icon_billboard = TRUE,
    get_icon = icon,
    get_icon_angle = 0,
    get_icon_color = "#000000ff",
    get_icon_pixel_offset = c(0, 0),
    get_icon_size = 1,
    text_size_max_pixels = 9007199254740991,
    text_size_min_pixels = 0,
    text_size_scale = 1,
    text_size_units = "pixels",
    text_background = FALSE,
    text_background_padding = c(0, 0, 0, 0),
    text_font_family = "Roboto, Helvetica, Arial, san-serif",
    text_font_weight = "normal",
    text_line_height = 1,
    text_max_width = -1,
    text_outline_color = "#000000ff",
    text_outline_width = 0,
    text_word_break = "break-word",
    text_billboard = TRUE,
```

```
  text_font_settings = list(),
  get_text = text,
  get_text_angle = 0,
  get_text_color = "#000000ff",
  get_text_pixel_offset = c(0, 0),
  get_text_size = 32,
  get_text_anchor = "middle",
  get_text_alignment_baseline = "center",
  get_text_background_color = "#ffffffff",
  get_text_border_color = "#000000ff",
  get_text_border_width = 0,
  line_joint_rounded = FALSE,
  line_cap_rounded = FALSE,
  line_miter_limit = 4,
  line_billboard = FALSE,
  extruded = FALSE,
  wireframe = FALSE,
  elevation_scale = 1,
  material = TRUE,
  get_elevation = 1000,
  point_type = "circle",
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_geojson_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  filled = cur_value(),
  stroked = cur_value(),
  line_width_max_pixels = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_scale = cur_value(),
  line_width_units = cur_value(),
  point_radius_max_pixels = cur_value(),
```

```
point_radius_min_pixels = cur_value(),
point_radius_scale = cur_value(),
point_radius_units = cur_value(),
point_antialiasing = cur_value(),
point_billboard = cur_value(),
get_fill_color = cur_value(),
get_line_color = cur_value(),
get_line_width = cur_value(),
get_point_radius = cur_value(),
icon_atlas = cur_value(),
icon_mapping = cur_value(),
icon_size_max_pixels = cur_value(),
icon_size_min_pixels = cur_value(),
icon_size_scale = cur_value(),
icon_size_units = cur_value(),
icon_alpha_cutoff = cur_value(),
icon_billboard = cur_value(),
get_icon = cur_value(),
get_icon_angle = cur_value(),
get_icon_color = cur_value(),
get_icon_pixel_offset = cur_value(),
get_icon_size = cur_value(),
text_size_max_pixels = cur_value(),
text_size_min_pixels = cur_value(),
text_size_scale = cur_value(),
text_size_units = cur_value(),
text_background = cur_value(),
text_background_padding = cur_value(),
text_font_family = cur_value(),
text_font_weight = cur_value(),
text_line_height = cur_value(),
text_max_width = cur_value(),
text_outline_color = cur_value(),
text_outline_width = cur_value(),
text_word_break = cur_value(),
text_billboard = cur_value(),
text_font_settings = cur_value(),
get_text = cur_value(),
get_text_angle = cur_value(),
get_text_color = cur_value(),
get_text_pixel_offset = cur_value(),
get_text_size = cur_value(),
get_text_anchor = cur_value(),
get_text_alignment_baseline = cur_value(),
get_text_background_color = cur_value(),
get_text_border_color = cur_value(),
get_text_border_width = cur_value(),
line_joint_rounded = cur_value(),
```

```
  line_cap_rounded = cur_value(),
  line_miter_limit = cur_value(),
  line_billboard = cur_value(),
  extruded = cur_value(),
  wireframe = cur_value(),
  elevation_scale = cur_value(),
  material = cur_value(),
  get_elevation = cur_value(),
  point_type = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL to geojson data that will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all

objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`filled` <boolean> If TRUE, draw the filled area of each point.

`stroked` <boolean> If TRUE, draw an outline around each object.

`line_width_max_pixels`
<number> The maximum line width in pixels.

`line_width_min_pixels`
<number> The minimum line width in pixels.

`line_width_scale`
<number> The line width multiplier.

`line_width_units`
<"pixels" | "common" | "meters"> The units of line width.

`point_radius_max_pixels`
<number> The maximum radius in pixels.

`point_radius_min_pixels`
<number> The minimum radius in pixels.

`point_radius_scale`
<number> The radius multiplier for all points.

`point_radius_units`
<"pixels" | "common" | "meters"> The units of point radius.

`point_antialiasing`
<boolean> If TRUE, circles are rendered with smoothed edges.

`point_billboard`
<boolean> If TRUE, circles always face the camera; if FALSE circles face up (z).

`get_fill_color` <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_color` <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_width` <accessor | scale | number> The outline of the object in units specified by `line_width_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`get_point_radius`
<accessor | scale | number> The radius of each point, in units specified by `radius_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`icon_atlas` <string|array> The image sprite containing raster icons. Must be either a string which is interpreted as a URL to an image, or an image bitmap.

`icon_mapping` <string|list> The image sprite index. Must be either a string which is interpreted as a URL to a json document, or a named-list of icon descriptors. See https://github.com/visgl/deck.gl/blob/master/docs/api-reference/layers/icon-layer.md#iconmapping-objectstring-optional for icon descriptor fields.

`icon_size_max_pixels`
<number> The maximum icon size in pixels.

icon_size_min_pixels
: <number> The minimum icon size in pixels.

icon_size_scale
: <number> The icon size multiplier.

icon_size_units
: <"pixels" | "common" | "meters"> The units of the icon size specified by get_icon_size.

icon_alpha_cutoff
: <number> Discard icon pixels whose opacity is below this threshold. A discarded pixel would create a "hole" in the icon that is not considered part of the object.

icon_billboard
: <boolean> If TRUE, the icon always faces the camera, otherwise it faces up (z).

get_icon
: <accessor> The name of the icon for each feature. Icon name must be present in the icon_mapping.

get_icon_angle
: <accessor | number> The rotating angle of each icon in degrees. Accepts a single numeric value, or a tidy-eval column of numbers.

get_icon_color
: <accessor | scale | color> The colour of each icon. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_icon_pixel_offset
: <accessor | number> The pixel offset for each icon. Accepts a single length-2 numeric vector, or a tidy-eval list column.

get_icon_size
: <accessor | scale | number> The size of each icon, in units specified by size_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

text_size_max_pixels
: <number> The maximum text label size in pixels.

text_size_min_pixels
: <number> The minimum text label size in pixels.

text_size_scale
: <number> The text label size multiplier.

text_size_units
: <"pixels" | "common" | "meters"> The units of the text label size, specified by get_text_size.

text_background
: <boolean> Whether to render background for text labels.

text_background_padding
: <numeric> The text background padding. Must be an array of 2 or 4 numbers.

text_font_family
: <string> Specifies a prioritised list of one or more font family names. See font-family.

text_font_weight
: <"normal" | "bold" | 100:900> The font weight. See font-weight

text_line_height
: <number> A unitless number that will be multiplied with get_size to set the line height.

text_max_width <number> Used together with text_word_break for wrapping text. Specifies the width limit to break the text into multiple lines.

text_outline_color
<color> The text outline colour. Requires text_font_settings$sdf = TRUE.

text_outline_width
<number> The text outline width, relative to font size. Requires text_font_settings$sdf = TRUE.

text_word_break
<"break-word" | "break-all"> Requires a valid text_max_width.

text_billboard <boolean> If TRUE, the text label always faces the camera, otherwise it faces up (z).

text_font_settings
<font_settings> Advanced options for fine tuning the appearance and performance of the generated font_atlas.

get_text <accessor> The text value of each text label. Accepts a tidy-eval character column of labels.

get_text_angle <accessor | number> The rotating angle of each text label in degrees. Accepts a single numeric value, or a tidy-eval column of numbers.

get_text_color <accessor | scale | color> The colour of each text label. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_text_pixel_offset
<accessor | number> The pixel offset for each text label. Accepts a single length-2 numeric vector, or a tidy-eval list column.

get_text_size <accessor | scale | number> The font size of each text label, in units specified by text_size_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_text_anchor
<accessor | "start" | "middle" | "end"> The text label anchor. May be a single value, or a tidy-eval character column.

get_text_alignment_baseline
<accessor | "top" | "center" | "bottom"> The text label alignment baseline. May be a single value, or a tidy-eval character column.

get_text_background_color
<accessor | scale | color> The text background colour, if text_background = TRUE. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_text_border_color
<accessor | scale | color> The text background border colour, if text_background = TRUE. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_text_border_width
<accessor | scale | number> The text background border width, if text_background = TRUE. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

line_joint_rounded
<boolean> If TRUE, draw round joints; else draw square joints.

line_cap_rounded
: <boolean> If TRUE, draw round caps; else draw square caps.

line_miter_limit
: number

line_billboard
: <boolean> If TRUE, extrude the path in screen space (width always faces) the camera; if FALSE, the width always faces up (z).

extruded
: <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

wireframe
: <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

elevation_scale
: <number> The elevation multiplier.

material
: <boolean>

get_elevation
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

point_type
: <"circle"|"icon"|"text"|combination> Determines how to render point and multipoint features. May be one of:

  - "circle"
  - "icon"
  - "text"

  Or a combination, separated by "+", e.g. "circle+text".

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/geojson-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

`gpu_grid_layer` *GPU Grid Layer*

Description

GPU Grid Layer

Usage

```
add_gpu_grid_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "GPUGridLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  color_domain = NULL,
 color_range = c("#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"),
  get_color_weight = 1,
  color_aggregation = "SUM",
  elevation_domain = NULL,
  elevation_range = c(0, 1000),
  get_elevation_weight = 1,
  elevation_aggregation = "SUM",
  elevation_scale = 1,
  cell_size = 1000,
  coverage = 1,
  get_position = position,
  extruded = FALSE,
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE
```

```
)

update_gpu_grid_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  color_domain = cur_value(),
  color_range = cur_value(),
  get_color_weight = cur_value(),
  color_aggregation = cur_value(),
  elevation_domain = cur_value(),
  elevation_range = cur_value(),
  get_elevation_weight = cur_value(),
  elevation_aggregation = cur_value(),
  elevation_scale = cur_value(),
  cell_size = cur_value(),
  coverage = cur_value(),
  get_position = cur_value(),
  extruded = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

data <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

visible <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

color_domain <number> The colour scale domain, default is set to the range of aggregated weights in each bin.

color_range <color> The colour palette. color_domain is divided into length(color_range) equal segments, each mapped to one color in color_range.

get_color_weight
<accessor | scale | number> The weight of each object used to calculate the colour value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

color_aggregation
<"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's colour.

elevation_domain
<number> The elevation scale input domain. Defaults to the range of the aggregated weights in each bin.

elevation_range
<number> The elevation scale output range.

get_elevation_weight
<accessor | scale | 'numeric> The weight of each object used to calculate the elevation value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

elevation_aggregation
<"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's elevation value.

elevation_scale
: <number> The elevation multiplier.

cell_size
: <number> The size of each cell in metres.

coverage
: <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by `coverage * radius`.

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

extruded
: <boolean> If `TRUE`, extrude objects along the z-axis; if `FALSE`, all objects will be flat.

material
: <boolean>

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - `normal`: Normal blending doesn't alter colours of overlapping objects.
  - `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/gpu-grid-layer.md

Other aggregation-layers: contour_layer, cpu_grid_layer, grid_layer, heatmap_layer, hexagon_layer, screen_grid_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

great_circle_layer *Great Circle Layer*

Description

Great Circle Layer

Usage

```
add_great_circle_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "GreatCircleLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_source_position = source_position,
  get_target_position = target_position,
  get_source_color = "#000000ff",
  get_target_color = "#000000ff",
  get_width = 1,
  get_height = 0,
  get_tilt = 0,
  great_circle = TRUE,
  width_units = "pixels",
  width_scale = 1,
  width_min_pixels = 0,
  width_max_pixels = 9007199254740991,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_great_circle_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
```

```
  get_source_position = cur_value(),
  get_target_position = cur_value(),
  get_source_color = cur_value(),
  get_target_color = cur_value(),
  get_width = cur_value(),
  get_height = cur_value(),
  get_tilt = cur_value(),
  great_circle = cur_value(),
  width_units = cur_value(),
  width_scale = cur_value(),
  width_min_pixels = cur_value(),
  width_max_pixels = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
: <accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

get_source_position
: <accessor> The feature source positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_target_position
: <accessor> The feature target positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_source_color
: <accessor | scale | color> The colour of the *source end* of the arc. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_target_color
: <accessor | scale | color> The colour of the *target end* of the arc. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_width
: <accessor | scale | number> The width of each object, in units specified by width_scale. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_height
: <accessor | scale | number> The multiplier of layer of layer height. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers. A value of 0 will make the layer flat.

get_tilt
: <accessor | number> Tilts the arcs by the specified number of degrees (between c(-90, 90)). Accepts a single numeric value or a tidy-eval column of numbers.

great_circle
: <boolean> If TRUE, create the arc along the shortest path on the earth surface.

width_units
: <"pixels" | "common" | "meters"> The units of the line_width.

width_scale
: <number> The scaling multiplier for the width of each line.

width_min_pixels
: <number> The minimum line width in pixels.

width_max_pixels
: <number> The maximum line width in pixels.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/great-circle-layer.md

Other geo-layers: h3_cluster_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, s2_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

grid_cell_layer *Grid Cell Layer*

Description

Grid Cell Layer

Usage

```
add_grid_cell_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "GridCellLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  disk_resolution = 20,
  vertices = NULL,
  radius = 1000,
  angle = 0,
  offset = c(1, 1),
  coverage = 1,
  elevation_scale = 1,
  radius_units = "meters",
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
```

```
  line_width_max_pixels = 9007199254740991,
  extruded = FALSE,
  wireframe = FALSE,
  filled = TRUE,
  stroked = FALSE,
  get_position = position,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  cell_size = 1000,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_grid_cell_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  disk_resolution = cur_value(),
  vertices = cur_value(),
  radius = cur_value(),
  angle = cur_value(),
  offset = cur_value(),
  coverage = cur_value(),
  elevation_scale = cur_value(),
  radius_units = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  extruded = cur_value(),
  wireframe = cur_value(),
  filled = cur_value(),
  stroked = cur_value(),
```

```
  get_position = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  cell_size = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`disk_resolution`
: <number> The number of sides to render the disk as. The disk is a regular polygon that fits inside the given `radius`. A higher resolution will yield a smoother look close-up, but also need more resources to render.

`vertices`
: `matrix` Replace the default geometry (regular polygon that fits inside the unit circle) with a custom one. The length of the array must be at least `disk_resolution`. Each vertex is a row `c(x, y)` that is the offset from the instance position, relative to the radius.

`radius`
: <number> The radius of the object in metres.

`angle`
: <number> The disk rotation, counter-clockwise in radians.

`offset`
: <number> The disk offset from the position, relative to the radius.

`coverage`
: <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by `coverage * radius`.

`elevation_scale`
: <number> The elevation multiplier.

`radius_units`
: <"pixels" | "common" | "meters"> The units of point radius.

`line_width_units`
: <"pixels" | "common" | "meters"> The units of line width.

`line_width_scale`
: <number> The line width multiplier.

`line_width_min_pixels`
: <number> The minimum line width in pixels.

`line_width_max_pixels`
: <number> The maximum line width in pixels.

`extruded`
: <boolean> If `TRUE`, extrude objects along the z-axis; if `FALSE`, all objects will be flat.

`wireframe`
: <boolean> If `TRUE` and `extruded == TRUE`, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

`filled`
: <boolean> If `TRUE`, draw the filled area of each point.

`stroked`
: <boolean> If `TRUE`, draw an outline around each object.

`get_position`
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_fill_color`
: <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_color`
: <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_width`
: <accessor | scale | number> The outline of the object in units specified by `line_width_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`get_elevation`
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

material <boolean>

cell_size <number> The size of each cell in metres.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle

<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/grid-cell-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

grid_layer *Grid Layer*

Description

Grid Layer

Usage

```
add_grid_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "GridLayer",
  group_name = NULL,
  data = NULL,
```

```
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  color_domain = NULL,
 color_range = c("#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"),
  get_color_weight = 1,
  color_aggregation = "SUM",
  elevation_domain = NULL,
  elevation_range = c(0, 1000),
  get_elevation_weight = 1,
  elevation_aggregation = "SUM",
  elevation_scale = 1,
  cell_size = 1000,
  coverage = 1,
  get_position = position,
  extruded = FALSE,
  material = TRUE,
  get_color_value = NULL,
  lower_percentile = 0,
  upper_percentile = 100,
  color_scale_type = "quantize",
  get_elevation_value = NULL,
  elevation_lower_percentile = 0,
  elevation_upper_percentile = 100,
  elevation_scale_type = "linear",
  gpu_aggregation = FALSE,
  blending_mode = "normal",
  visibility_toggle = TRUE
)

update_grid_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
```

```
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  color_domain = cur_value(),
  color_range = cur_value(),
  get_color_weight = cur_value(),
  color_aggregation = cur_value(),
  elevation_domain = cur_value(),
  elevation_range = cur_value(),
  get_elevation_weight = cur_value(),
  elevation_aggregation = cur_value(),
  elevation_scale = cur_value(),
  cell_size = cur_value(),
  coverage = cur_value(),
  get_position = cur_value(),
  extruded = cur_value(),
  material = cur_value(),
  get_color_value = cur_value(),
  lower_percentile = cur_value(),
  upper_percentile = cur_value(),
  color_scale_type = cur_value(),
  get_elevation_value = cur_value(),
  elevation_lower_percentile = cur_value(),
  elevation_upper_percentile = cur_value(),
  elevation_scale_type = cur_value(),
  gpu_aggregation = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

color_domain <number> The colour scale domain, default is set to the range of aggregated weights in each bin.

color_range <color> The colour palette. color_domain is divided into length(color_range) equal segments, each mapped to one color in color_range.

get_color_weight
<accessor | scale | number> The weight of each object used to calculate the colour value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

color_aggregation
<"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's colour.

elevation_domain
<number> The elevation scale input domain. Defaults to the range of the aggregated weights in each bin.

elevation_range
<number> The elevation scale output range.

get_elevation_weight
<accessor | scale | 'numeric> The weight of each object used to calculate the elevation value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

elevation_aggregation
<"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's elevation value.

elevation_scale
<number> The elevation multiplier.

cell_size <number> The size of each cell in metres.

coverage <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by coverage * radius.

get_position <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

extruded <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

material <boolean>

get_color_value

<JS> After data objects are aggregated into bins, this accessor is called on each bin to get the value that its colour is based on. If supplied, this will override the effect of get_color_weight and color_aggregation props. See deck.gl docs for details.

lower_percentile

<number> Between 0 and 100. Filter bins and re-calculate colour by lower_percentile. Cells with value < lower_percentile will be hidden.

upper_percentile

<number> Between 0 and 100. Filter bins and re-calculate colour by upper_percentile. Cells with value < upper_percentile will be hidden.

color_scale_type

<"quantize"|"linear"|"quantile"|"ordinal"> The scaling function used to determine the colour of a the grid cell.

get_elevation_value

<JS> After data objects are aggregated into bins, this accessor is called on each bin to get the value that its elevation is based on. If supplied, this will override the effect of get_elevation_weight and elevation_aggregation props. See deck.gl docs for details.

elevation_lower_percentile

<number> Between 0 and 100. Filter bins and re-calculate elevation by elevation_lower_percentile. Cells with value < elevation_lower_percentile will be hidden.

elevation_upper_percentile

<number> Between 0 and 100. Filter bins and re-calculate elevation by elevation_upper_percentile. Cells with value < elevation_upper_percentile will be hidden.

elevation_scale_type

<"quantize"|"linear"|"quantile"|"ordinal"> The scaling function used to determine the elevation of a the grid cell.

gpu_aggregation

<boolean> If TRUE, aggregation is performed on GPU if supported. Requires WebGL2.

blending_mode <"normal"|"additive"|"subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle

<boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/grid-layer.md

Other aggregation-layers: contour_layer, cpu_grid_layer, gpu_grid_layer, heatmap_layer, hexagon_layer, screen_grid_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

h3_cluster_layer *H3 Cluster Layer*

Description

H3 Cluster Layer

Usage

```
add_h3_cluster_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "H3ClusterLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  stroked = TRUE,
  filled = TRUE,
  extruded = FALSE,
  elevation_scale = 1,
  wireframe = FALSE,
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  line_joint_rounded = FALSE,
```

```
  line_miter_limit = 4,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  get_hexagons = hexagons,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_h3_cluster_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  stroked = cur_value(),
  filled = cur_value(),
  extruded = cur_value(),
  elevation_scale = cur_value(),
  wireframe = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  line_joint_rounded = cur_value(),
  line_miter_limit = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  get_hexagons = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
stroked	<boolean> If TRUE, draw an outline around each object.
filled	<boolean> If TRUE, draw the filled area of each point.
extruded	<boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.
elevation_scale	<number> The elevation multiplier.
wireframe	<boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

`line_width_units`
: <"pixels" | "common" | "meters"> The units of line width.

`line_width_scale`
: <number> The line width multiplier.

`line_width_min_pixels`
: <number> The minimum line width in pixels.

`line_width_max_pixels`
: <number> The maximum line width in pixels.

`line_joint_rounded`
: <boolean>

`line_miter_limit`
: number

`get_fill_color`
: <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_color`
: <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_width`
: <accessor | scale | number> The outline of the object in units specified by `line_width_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`get_elevation`
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`material`
: <boolean>

`get_hexagons`
: <accessor>

`blending_mode`
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:
  - `normal`: Normal blending doesn't alter colours of overlapping objects.
  - `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

`visibility_toggle`
: <boolean> Whether this layer will appear in the layer selector.

`tooltip`
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if `pickable == TRUE`. Supports tidy-select if a `data` is a `data.frame`. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/h3-cluster-layer.md

Other geo-layers: great_circle_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, s2_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_hexagon_layer,

heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

h3_hexagon_layer *H3 Hexagon Layer*

Description

H3 Hexagon Layer

Usage

```
add_h3_hexagon_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "H3HexagonLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  stroked = TRUE,
  filled = TRUE,
  extruded = FALSE,
  elevation_scale = 1,
  wireframe = FALSE,
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  line_joint_rounded = FALSE,
  line_miter_limit = 4,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  high_precision = "auto",
  coverage = 1,
```

```
  center_hexagon = NULL,
  get_hexagon = hexagon,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_h3_hexagon_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  stroked = cur_value(),
  filled = cur_value(),
  extruded = cur_value(),
  elevation_scale = cur_value(),
  wireframe = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  line_joint_rounded = cur_value(),
  line_miter_limit = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  high_precision = cur_value(),
  coverage = cur_value(),
  center_hexagon = cur_value(),
  get_hexagon = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
stroked	<boolean> If TRUE, draw an outline around each object.
filled	<boolean> If TRUE, draw the filled area of each point.
extruded	<boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.
elevation_scale	<number> The elevation multiplier.
wireframe	<boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

line_width_units
: <"pixels" | "common" | "meters"> The units of line width.

line_width_scale
: <number> The line width multiplier.

line_width_min_pixels
: <number> The minimum line width in pixels.

line_width_max_pixels
: <number> The maximum line width in pixels.

line_joint_rounded
: <boolean>

line_miter_limit
: number

get_fill_color
: <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_color
: <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_width
: <accessor | scale | number> The outline of the object in units specified by `line_width_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_elevation
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

material
: <boolean>

high_precision
: <boolean> If `TRUE`, draw each hexagon as a polygon using each hexagon's true geometry. If `FALSE`, draw each hexagon with the same shape as `center_hexagon`. High precision trades rendering performance for geometry precision. High precision is required (forcibly enabled) when:
  - Rendering hex resolutions in range `c(0, 5)`
  - Pentagons are present in the data
  - Multiple resolution hexes are present in the data

coverage
: <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by `coverage * radius`.

center_hexagon
: <string> If `high_precision == FALSE`, defines the H3 hexagon that will be used as the shape of all hexagons in the layer. Defaults to the hexagon in the centre of the viewport.

get_hexagon
: <accessor> The H3 hex token for each H3 hexagon. Accepts a tidy-eval character column.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:
  - `normal`: Normal blending doesn't alter colours of overlapping objects.
  - `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/h3-hexagon-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, mvt_layer, quadkey_layer, s2_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

heatmap_layer *Heatmap Layer*

Description

Heatmap Layer

Usage

```
add_heatmap_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "HeatmapLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_position = position,
  get_weight = 1,
  intensity = 1,
```

```
  radius_pixels = 50,
 color_range = c("#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"),
  threshold = 0.05,
  color_domain = NULL,
  aggregation = "SUM",
  weights_texture_size = 2048,
  debounce_timeout = 500,
  blending_mode = "normal",
  visibility_toggle = TRUE
)

update_heatmap_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_position = cur_value(),
  get_weight = cur_value(),
  intensity = cur_value(),
  radius_pixels = cur_value(),
  color_range = cur_value(),
  threshold = cur_value(),
  color_domain = cur_value(),
  aggregation = cur_value(),
  weights_texture_size = cur_value(),
  debounce_timeout = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly

defined for updatable layers in a shiny application.

`name` <string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

`group_name` <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if `visibility_toggle = TRUE`.

`data` <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

`visible` <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

`pickable` <boolean> Determines if the layer responds to pointer / touch events.

`opacity` <number> Determines the layer's opacity.

`wrap_longitude` <boolean> Normalises geometry longitudes.

`position_format`

<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`

<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`get_position` <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_weight` <accessor | scale | number> The weight of each object. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`intensity` <number> Value that is multiplied with the total weight at a pixel to obtain the final weight. A value > 1 biases the output colour towards the higher end of the `color_range`, and a value < 1 biases the output towards the lower end of the `color_range`.

`radius_pixels` <number> The radius of the circle in pixels.

`color_range` <color> The colour palette. `color_domain` is divided into `length(color_range)` equal segments, each mapped to one color in `color_range`.

`threshold` <number> Larger threshold values creates smoother boundaries of colour *blobs*, while making pixels with low weight values more transparent. Ignored when `color_domain` is specified.

`color_domain` <number> The colour scale domain, default is set to the range of aggregated weights in each bin.

`aggregation` <"SUM" | "MEAN" | "MIN" | "MAX"> Defines the aggregation function.

`weights_texture_size`
: <number> The size of the weight texture. Smaller texture sizes can improve rendering performance, but lead to visible pixelation.

`debounce_timeout`
: <number> Debounce interval (milliseconds) for triggering aggregation.

`blending_mode`
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:
  - `normal`: Normal blending doesn't alter colours of overlapping objects.
  - `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

`visibility_toggle`
: <boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/heatmap-layer.md

Other aggregation-layers: contour_layer, cpu_grid_layer, gpu_grid_layer, grid_layer, hexagon_layer, screen_grid_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

hexagon_layer *Hexagon Layer*

Description

Hexagon Layer

Usage

```
add_hexagon_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "HexagonLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
```

```
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  color_domain = NULL,
 color_range = c("#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"),
  get_color_value = NULL,
  get_color_weight = 1,
  color_aggregation = "SUM",
  lower_percentile = 0,
  upper_percentile = 100,
  color_scale_type = "quantize",
  elevation_domain = NULL,
  elevation_range = c(0, 1000),
  get_elevation_value = NULL,
  get_elevation_weight = 1,
  elevation_aggregation = "SUM",
  elevation_lower_percentile = 0,
  elevation_upper_percentile = 100,
  elevation_scale = 1,
  elevation_scale_type = "linear",
  radius = 1000,
  coverage = 1,
  extruded = FALSE,
  get_position = position,
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE
)

update_hexagon_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
```

```
  color_domain = cur_value(),
  color_range = cur_value(),
  get_color_value = cur_value(),
  get_color_weight = cur_value(),
  color_aggregation = cur_value(),
  lower_percentile = cur_value(),
  upper_percentile = cur_value(),
  color_scale_type = cur_value(),
  elevation_domain = cur_value(),
  elevation_range = cur_value(),
  get_elevation_value = cur_value(),
  get_elevation_weight = cur_value(),
  elevation_aggregation = cur_value(),
  elevation_lower_percentile = cur_value(),
  elevation_upper_percentile = cur_value(),
  elevation_scale = cur_value(),
  elevation_scale_type = cur_value(),
  radius = cur_value(),
  coverage = cur_value(),
  extruded = cur_value(),
  get_position = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.

`wrap_longitude` <boolean> Normalises geometry longitudes.

`position_format`
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`
<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`color_domain` <number> The colour scale domain, default is set to the range of aggregated weights in each bin.

`color_range` <color> The colour palette. `color_domain` is divided into `length(color_range)` equal segments, each mapped to one color in `color_range`.

`get_color_value`
<JS> After data objects are aggregated into bins, this accessor is called on each bin to get the value that its colour is based on. If supplied, this will override the effect of `get_color_weight` and `color_aggregation` props. See deck.gl docs for details.

`get_color_weight`
<accessor | scale | number> The weight of each object used to calculate the colour value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`color_aggregation`
<"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's colour.

`lower_percentile`
<number> Between `0` and `100`. Filter bins and re-calculate colour by `lower_percentile`. Cells with `value` < `lower_percentile` will be hidden.

`upper_percentile`
<number> Between `0` and `100`. Filter bins and re-calculate colour by `upper_percentile`. Cells with `value` < `upper_percentile` will be hidden.

`color_scale_type`
<"quantize" | "linear" | "quantile" | "ordinal"> The scaling function used to determine the colour of a the grid cell.

`elevation_domain`
<number> The elevation scale input domain. Defaults to the range of the aggregated weights in each bin.

`elevation_range`
<number> The elevation scale output range.

get_elevation_value
: <JS> After data objects are aggregated into bins, this accessor is called on each bin to get the value that its elevation is based on. If supplied, this will override the effect of `get_elevation_weight` and `elevation_aggregation` props. See deck.gl docs for details.

get_elevation_weight
: <accessor | scale | 'numeric> The weight of each object used to calculate the elevation value for a bin. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

elevation_aggregation
: <"SUM" | "MEAN" | "MIN" | "MAX"> Operation used to aggregate data values/weights to calculate a bin's elevation value.

elevation_lower_percentile
: <number> Between `0` and `100`. Filter bins and re-calculate elevation by `elevation_lower_percentile`. Cells with `value` < `elevation_lower_percentile` will be hidden.

elevation_upper_percentile
: <number> Between `0` and `100`. Filter bins and re-calculate elevation by `elevation_upper_percentile`. Cells with `value` < `elevation_upper_percentile` will be hidden.

elevation_scale
: <number> The elevation multiplier.

elevation_scale_type
: <"quantize" | "linear" | "quantile" | "ordinal"> The scaling function used to determine the elevation of a the grid cell.

radius
: <number> The radius of the hexagon bin in metres.

coverage
: <number> Radius multiplier, between 0 - 1. The radius of each disk is calculated by `coverage * radius`.

extruded
: <boolean> If `TRUE`, extrude objects along the z-axis; if `FALSE`, all objects will be flat.

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

material
: <boolean>

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - `normal`: Normal blending doesn't alter colours of overlapping objects.
  - `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/hexagon-layer.md

Other aggregation-layers: contour_layer, cpu_grid_layer, gpu_grid_layer, grid_layer, heatmap_layer, screen_grid_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

`icon_layer` *Icon Layer*

Description

Icon Layer

Usage

```
add_icon_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "IconLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  icon_atlas = NULL,
  icon_mapping = list(),
  size_scale = 1,
  billboard = TRUE,
  size_units = "pixels",
  size_min_pixels = 0,
  size_max_pixels = 9007199254740991,
  alpha_cutoff = 0.05,
  get_position = position,
  get_icon = icon,
  get_color = "#000000ff",
  get_size = 1,
  get_angle = 0,
  get_pixel_offset = c(0, 0),
```

```
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_icon_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  icon_atlas = cur_value(),
  icon_mapping = cur_value(),
  size_scale = cur_value(),
  billboard = cur_value(),
  size_units = cur_value(),
  size_min_pixels = cur_value(),
  size_max_pixels = cur_value(),
  alpha_cutoff = cur_value(),
  get_position = cur_value(),
  get_icon = cur_value(),
  get_color = cur_value(),
  get_size = cur_value(),
  get_angle = cur_value(),
  get_pixel_offset = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.

name <string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

group_name <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

data <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

visible <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format <"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color <accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

icon_atlas <string|array> The image sprite containing raster icons. Must be either a string which is interpreted as a URL to an image, or an image bitmap.

icon_mapping <string|list> The image sprite index. Must be either a string which is interpreted as a URL to a json document, or a named-list of icon descriptors. See https://github.com/visgl/deck.gl/blob/master/docs/api-reference/layers/icon-layer.md#iconmapping-objectstring-optional for icon descriptor fields.

size_scale <number> The size multiplier.

billboard <boolean> If TRUE, the icon always faces the camera, otherwise it faces up (z).

size_units <"pixels" | "common" | "meters"> The units of the size specified by get_size.

size_min_pixels <number> The minimum size in pixels.

size_max_pixels <number> The maximum size in pixels.

alpha_cutoff <number> Discard pixels whose opacity is below this threshold. A discarded pixel would create a "hole" in the icon that is not considered part of the object.

get_position <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_icon <accessor> The name of the icon for each feature. Icon name must be present in the icon_mapping.

get_color <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_size <accessor | scale | number> The size of each icon, in units specified by size_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_angle <accessor | number> The rotating angle of each icon in degrees. Accepts a single numeric value, or a tidy-eval column of numbers.

get_pixel_offset
<accessor | number> The pixel offset for each object. Accepts a single length-2 numeric vector, or a tidy-eval list column.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/icon-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, grid_cell_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

layers *Layers*

Description

Get map layers

Usage

```
layers(rdeck)
```

Arguments

rdeck	an rdeck instance

line_layer *Line Layer*

Description

Line Layer

Usage

```
add_line_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "LineLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_source_position = source_position,
  get_target_position = target_position,
  get_color = "#000000ff",
  get_width = 1,
  width_units = "pixels",
  width_scale = 1,
```

```
  width_min_pixels = 0,
  width_max_pixels = 9007199254740991,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_line_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_source_position = cur_value(),
  get_target_position = cur_value(),
  get_color = cur_value(),
  get_width = cur_value(),
  width_units = cur_value(),
  width_scale = cur_value(),
  width_min_pixels = cur_value(),
  width_max_pixels = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

`data` <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

`visible` <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

`pickable` <boolean> Determines if the layer responds to pointer / touch events.

`opacity` <number> Determines the layer's opacity.

`wrap_longitude` <boolean> Normalises geometry longitudes.

`position_format`
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`
<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`get_source_position`
<accessor> The feature source positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_target_position`
<accessor> The feature target positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_color` <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_width` <accessor | scale | number> The width of each object, in units specified by `width_scale`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`width_units` <"pixels" | "common" | "meters"> The units of the `line_width`.

`width_scale` <number> The scaling multiplier for the width of each line.

`width_min_pixels`
<number> The minimum line width in pixels.

`width_max_pixels`
<number> The maximum line width in pixels.

`blending_mode` <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- `normal`: Normal blending doesn't alter colours of overlapping objects.
- `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.

- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/line-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, grid_cell_layer, icon_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

log_trans *Log transformation*

Description

Applies a log transform on the input: y = log(abs(x), b), where:

- x is the input vector
- b is the log base

Usage

```
log_trans(base = exp(1))
```

Arguments

base
: <number> The log base

Details

If x is negative, the result is multiplied by -1.

This transform requires that the input range doesn't cross zero. Transforming an input that crosses 0 will succeed and give predictable output, but its inverse will not, due to unsigned 0 (i.e. -1 * log(1) == log(1)).

See Also

Other transform: `power_trans()`, `symlog_trans()`

`mapbox_access_token` *Mapbox access token*

Description

A mapbox access token is required for rendering the mapbox basemap (regardless of tiles used) and mapbox services (tiles). To use a basemap, you need to register for a mapbox account. Mapbox has a generous free tier.

Each rdeck map *rendered* equates to a map load for web.

Usage

```
mapbox_access_token()
```

Details

The mapbox token is read from the following locations (in order):

- `getOption("rdeck.mapbox_access_token")`
- `Sys.getenv("MAPBOX_ACCESS_TOKEN")`
- `Sys.getenv("MAPBOX_TOKEN")`

See Also

https://docs.mapbox.com/help/glossary/access-token

`mvt_layer` *MVT Layer*

Description

MVT Layer

Usage

```
add_mvt_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "MVTLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_tile_data = NULL,
  extent = NULL,
  tile_size = 512,
  max_zoom = NULL,
  min_zoom = 0,
  max_cache_size = NULL,
  max_cache_byte_size = NULL,
  refinement_strategy = "best-available",
  z_range = NULL,
  max_requests = 6,
  zoom_offset = 0,
  filled = TRUE,
  stroked = TRUE,
  line_width_max_pixels = 9007199254740991,
  line_width_min_pixels = 0,
  line_width_scale = 1,
  line_width_units = "meters",
  point_radius_max_pixels = 9007199254740991,
  point_radius_min_pixels = 0,
  point_radius_scale = 1,
  point_radius_units = "meters",
  point_antialiasing = TRUE,
  point_billboard = FALSE,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_point_radius = 1,
  icon_atlas = NULL,
  icon_mapping = list(),
  icon_size_max_pixels = 9007199254740991,
  icon_size_min_pixels = 0,
  icon_size_scale = 1,
```

```
  icon_size_units = "pixels",
  icon_alpha_cutoff = 0.05,
  icon_billboard = TRUE,
  get_icon = icon,
  get_icon_angle = 0,
  get_icon_color = "#000000ff",
  get_icon_pixel_offset = c(0, 0),
  get_icon_size = 1,
  text_size_max_pixels = 9007199254740991,
  text_size_min_pixels = 0,
  text_size_scale = 1,
  text_size_units = "pixels",
  text_background = FALSE,
  text_background_padding = c(0, 0, 0, 0),
  text_font_family = "Roboto, Helvetica, Arial, san-serif",
  text_font_weight = "normal",
  text_line_height = 1,
  text_max_width = -1,
  text_outline_color = "#000000ff",
  text_outline_width = 0,
  text_word_break = "break-word",
  text_billboard = TRUE,
  text_font_settings = list(),
  get_text = text,
  get_text_angle = 0,
  get_text_color = "#000000ff",
  get_text_pixel_offset = c(0, 0),
  get_text_size = 32,
  get_text_anchor = "middle",
  get_text_alignment_baseline = "center",
  get_text_background_color = "#ffffffff",
  get_text_border_color = "#000000ff",
  get_text_border_width = 0,
  line_joint_rounded = FALSE,
  line_cap_rounded = FALSE,
  line_miter_limit = 4,
  line_billboard = FALSE,
  extruded = FALSE,
  wireframe = FALSE,
  elevation_scale = 1,
  material = TRUE,
  get_elevation = 1000,
  point_type = "circle",
  unique_id_property = "",
  highlighted_feature_id = NULL,
  binary = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE,
```

```
  tooltip = NULL
)

update_mvt_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_tile_data = cur_value(),
  extent = cur_value(),
  tile_size = cur_value(),
  max_zoom = cur_value(),
  min_zoom = cur_value(),
  max_cache_size = cur_value(),
  max_cache_byte_size = cur_value(),
  refinement_strategy = cur_value(),
  z_range = cur_value(),
  max_requests = cur_value(),
  zoom_offset = cur_value(),
  filled = cur_value(),
  stroked = cur_value(),
  line_width_max_pixels = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_scale = cur_value(),
  line_width_units = cur_value(),
  point_radius_max_pixels = cur_value(),
  point_radius_min_pixels = cur_value(),
  point_radius_scale = cur_value(),
  point_radius_units = cur_value(),
  point_antialiasing = cur_value(),
  point_billboard = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_point_radius = cur_value(),
  icon_atlas = cur_value(),
  icon_mapping = cur_value(),
  icon_size_max_pixels = cur_value(),
```

```
    icon_size_min_pixels = cur_value(),
    icon_size_scale = cur_value(),
    icon_size_units = cur_value(),
    icon_alpha_cutoff = cur_value(),
    icon_billboard = cur_value(),
    get_icon = cur_value(),
    get_icon_angle = cur_value(),
    get_icon_color = cur_value(),
    get_icon_pixel_offset = cur_value(),
    get_icon_size = cur_value(),
    text_size_max_pixels = cur_value(),
    text_size_min_pixels = cur_value(),
    text_size_scale = cur_value(),
    text_size_units = cur_value(),
    text_background = cur_value(),
    text_background_padding = cur_value(),
    text_font_family = cur_value(),
    text_font_weight = cur_value(),
    text_line_height = cur_value(),
    text_max_width = cur_value(),
    text_outline_color = cur_value(),
    text_outline_width = cur_value(),
    text_word_break = cur_value(),
    text_billboard = cur_value(),
    text_font_settings = cur_value(),
    get_text = cur_value(),
    get_text_angle = cur_value(),
    get_text_color = cur_value(),
    get_text_pixel_offset = cur_value(),
    get_text_size = cur_value(),
    get_text_anchor = cur_value(),
    get_text_alignment_baseline = cur_value(),
    get_text_background_color = cur_value(),
    get_text_border_color = cur_value(),
    get_text_border_width = cur_value(),
    line_joint_rounded = cur_value(),
    line_cap_rounded = cur_value(),
    line_miter_limit = cur_value(),
    line_billboard = cur_value(),
    extruded = cur_value(),
    wireframe = cur_value(),
    elevation_scale = cur_value(),
    material = cur_value(),
    get_elevation = cur_value(),
    point_type = cur_value(),
    unique_id_property = cur_value(),
    highlighted_feature_id = cur_value(),
    binary = cur_value(),
```

```
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<character | tile_json> Defines the remote data for the layer. May take any of: • A tile_json() object • A string which is a valid URL to a tilejson • A character vector of tile url templates, each containing placeholders for tile coordinates: {z}/{x}/{y}
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
get_tile_data	<JS> retrieves the data of each tile. See deck.gl getTileData.

`extent` <c(min_x, min_y, max_x, max_y)> Tiles in this bounding box will be rendered at `min_zoom`, when zoomed out below `min_zoom`.

`tile_size` <number> A power of 2 that is the pixel dimensions of the tile.

`max_zoom` <number> Tiles above this zoom level are not shown. Defaults to `NULL`.

`min_zoom` <number> Tiles below this zoom level are not shown. Defaults to `0`.

`max_cache_size` <number> Maximum number of tiles that can be cached. Defaults to 5x the number of tiles in current viewport.

`max_cache_byte_size`
<number> Maximum memory used for caching tiles.

`refinement_strategy`
<"best-available" | "no-overlap" | "never"> How the tile layer refines visibility of tiles. Defaults to "best-available".

`z_range` <c(min, max)> Array representing the range of heights in the tile.

`max_requests` <number> Maximum number of concurrent HTTP requests across all specified tile provider domains. If a negative number is supplied no throttling occurs (HTTP/2 only).

`zoom_offset` <int> The offset changes the zoom level at which the tiles are fetched.

`filled` <boolean> If `TRUE`, draw the filled area of each point.

`stroked` <boolean> If `TRUE`, draw an outline around each object.

`line_width_max_pixels`
<number> The maximum line width in pixels.

`line_width_min_pixels`
<number> The minimum line width in pixels.

`line_width_scale`
<number> The line width multiplier.

`line_width_units`
<"pixels" | "common" | "meters"> The units of line width.

`point_radius_max_pixels`
<number> The maximum radius in pixels.

`point_radius_min_pixels`
<number> The minimum radius in pixels.

`point_radius_scale`
<number> The radius multiplier for all points.

`point_radius_units`
<"pixels" | "common" | "meters"> The units of point radius.

`point_antialiasing`
<boolean> If `TRUE`, circles are rendered with smoothed edges.

`point_billboard`
<boolean> If `TRUE`, circles always face the camera; if `FALSE` circles face up (z).

`get_fill_color` <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_color` <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_width <accessor | scale | number> The outline of the object in units specified by line_width_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_point_radius
<accessor | scale | number> The radius of each point, in units specified by radius_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

icon_atlas <string|array> The image sprite containing raster icons. Must be either a string which is interpreted as a URL to an image, or an image bitmap.

icon_mapping <string|list> The image sprite index. Must be either a string which is interpreted as a URL to a json document, or a named-list of icon descriptors. See https://github.com/visgl/deck.gl/blob/master/docs/api-reference/layers/icon-layer.md#iconmapping-objectstring-optional for icon descriptor fields.

icon_size_max_pixels
<number> The maximum icon size in pixels.

icon_size_min_pixels
<number> The minimum icon size in pixels.

icon_size_scale
<number> The icon size multiplier.

icon_size_units
<"pixels" | "common" | "meters"> The units of the icon size specified by get_icon_size.

icon_alpha_cutoff
<number> Discard icon pixels whose opacity is below this threshold. A discarded pixel would create a "hole" in the icon that is not considered part of the object.

icon_billboard <boolean> If TRUE, the icon always faces the camera, otherwise it faces up (z).

get_icon <accessor> The name of the icon for each feature. Icon name must be present in the icon_mapping.

get_icon_angle <accessor | number> The rotating angle of each icon in degrees. Accepts a single numeric value, or a tidy-eval column of numbers.

get_icon_color <accessor | scale | color> The colour of each icon. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_icon_pixel_offset
<accessor | number> The pixel offset for each icon. Accepts a single length-2 numeric vector, or a tidy-eval list column.

get_icon_size <accessor | scale | number> The size of each icon, in units specified by size_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

text_size_max_pixels
<number> The maximum text label size in pixels.

text_size_min_pixels
<number> The minimum text label size in pixels.

text_size_scale
<number> The text label size multiplier.

text_size_units
: <"pixels" | "common" | "meters"> The units of the text label size, specified by get_text_size.

text_background
: <boolean> Whether to render background for text labels.

text_background_padding
: <numeric> The text background padding. Must be an array of 2 or 4 numbers.

text_font_family
: <string> Specifies a prioritised list of one or more font family names. See font-family.

text_font_weight
: <"normal" | "bold" | 100:900> The font weight. See font-weight

text_line_height
: <number> A unitless number that will be multiplied with get_size to set the line height.

text_max_width
: <number> Used together with text_word_break for wrapping text. Specifies the width limit to break the text into multiple lines.

text_outline_color
: <color> The text outline colour. Requires text_font_settings$sdf = TRUE.

text_outline_width
: <number> The text outline width, relative to font size. Requires text_font_settings$sdf = TRUE.

text_word_break
: <"break-word" | "break-all"> Requires a valid text_max_width.

text_billboard
: <boolean> If TRUE, the text label always faces the camera, otherwise it faces up (z).

text_font_settings
: <font_settings> Advanced options for fine tuning the appearance and performance of the generated font_atlas.

get_text
: <accessor> The text value of each text label. Accepts a tidy-eval character column of labels.

get_text_angle
: <accessor | number> The rotating angle of each text label in degrees. Accepts a single numeric value, or a tidy-eval column of numbers.

get_text_color
: <accessor | scale | color> The colour of each text label. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_text_pixel_offset
: <accessor | number> The pixel offset for each text label. Accepts a single length-2 numeric vector, or a tidy-eval list column.

get_text_size
: <accessor | scale | number> The font size of each text label, in units specified by text_size_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_text_anchor
: <accessor | "start" | "middle" | "end"> The text label anchor. May be a single value, or a tidy-eval character column.

get_text_alignment_baseline
: <accessor | "top" | "center" | "bottom"> The text label alignment baseline. May be a single value, or a tidy-eval character column.

get_text_background_color
: <accessor | scale | color> The text background colour, if text_background = TRUE. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_text_border_color
: <accessor | scale | color> The text background border colour, if text_background = TRUE. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_text_border_width
: <accessor | scale | number> The text background border width, if text_background = TRUE. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

line_joint_rounded
: <boolean>

line_cap_rounded
: <boolean> If TRUE, draw round caps; else draw square caps.

line_miter_limit
: number

line_billboard
: <boolean> If TRUE, extrude the path in screen space (width always faces) the camera; if FALSE, the width always faces up (z).

extruded
: <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

wireframe
: <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

elevation_scale
: <number> The elevation multiplier.

material
: <boolean>

get_elevation
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

point_type
: <"circle"|"icon"|"text"|combination> Determines how to render point and multipoint features. May be one of:

  - "circle"
  - "icon"
  - "text"

  Or a combination, separated by "+", e.g. "circle+text".

unique_id_property
: <string> Used for highlighting features across tiles. Features on separate tiles are deemed to be *the same feature* by the supplied property name. If no value is supplied, the feature id will be used.

highlighted_feature_id
: <number | string> When provided, a feature with ID corresponding to the supplied value with be highlighted with highlight_color.

binary
: <boolean> Improves rendering performance by removing the tile serialisation and deserialisation between worker and main thread.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/mvt-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, quadkey_layer, s2_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

path_layer *Path Layer*

Description

Path Layer

Usage

```
add_path_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "PathLayer",
```

```
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  width_units = "meters",
  width_scale = 1,
  width_min_pixels = 0,
  width_max_pixels = 9007199254740991,
  joint_rounded = FALSE,
  cap_rounded = FALSE,
  miter_limit = 4,
  billboard = FALSE,
  get_path = path,
  get_color = "#000000ff",
  get_width = 1,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_path_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  width_units = cur_value(),
  width_scale = cur_value(),
  width_min_pixels = cur_value(),
  width_max_pixels = cur_value(),
  joint_rounded = cur_value(),
  cap_rounded = cur_value(),
  miter_limit = cur_value(),
```

```
  billboard = cur_value(),
  get_path = cur_value(),
  get_color = cur_value(),
  get_width = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
width_units	<"pixels" | "common" | "meters"> The units of the line_width.

`width_scale` <number> The scaling multiplier for the width of each line.

`width_min_pixels` <number> The minimum line width in pixels.

`width_max_pixels` <number> The maximum line width in pixels.

`joint_rounded` <boolean> If `TRUE`, draw round joints; else draw square joints.

`cap_rounded` <boolean> If `TRUE`, draw round caps; else draw square caps.

`miter_limit` <number> The maximum extent of a joint in ratio to the stroke width. Only applicable if `rounded == FALSE`.

`billboard` <boolean> If `TRUE`, extrude the path in screen space (width always faces) the camera; if `FALSE`, the width always faces up (z).

`get_path` <accessor> The feature paths. A <linestring/multilinestring> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_color` <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_width` <accessor | scale | number> The width of each object, in units specified by `width_scale`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`blending_mode` <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- `normal`: Normal blending doesn't alter colours of overlapping objects.
- `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

`visibility_toggle` <boolean> Whether this layer will appear in the layer selector.

`tooltip` <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if `pickable == TRUE`. Supports tidy-select if a `data` is a `data.frame`. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/path-layer.md

Other core-layers: `arc_layer`, `bitmap_layer`, `column_layer`, `geojson_layer`, `grid_cell_layer`, `icon_layer`, `line_layer`, `point_cloud_layer`, `polygon_layer`, `scatterplot_layer`, `solid_polygon_layer`, `text_layer`

Other layers: `arc_layer`, `bitmap_layer`, `column_layer`, `contour_layer`, `cpu_grid_layer`, `geojson_layer`, `gpu_grid_layer`, `great_circle_layer`, `grid_cell_layer`, `grid_layer`, `h3_cluster_layer`, `h3_hexagon_layer`, `heatmap_layer`, `hexagon_layer`, `icon_layer`, `line_layer`, `mvt_layer`, `point_cloud_layer`, `polygon_layer`, `quadkey_layer`, `s2_layer`, `scatterplot_layer`, `scenegraph_layer`, `screen_grid_layer`, `simple_mesh_layer`, `solid_polygon_layer`, `terrain_layer`, `text_layer`, `tile_3d_layer`, `tile_layer`, `trips_layer`

point_cloud_layer *Point Cloud Layer*

Description

Point Cloud Layer

Usage

```
add_point_cloud_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "PointCloudLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  size_units = "pixels",
  point_size = 10,
  get_position = position,
  get_normal = c(0, 0, 1),
  get_color = "#000000ff",
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_point_cloud_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
```

```
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  size_units = cur_value(),
  point_size = cur_value(),
  get_position = cur_value(),
  get_normal = cur_value(),
  get_color = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all

objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`size_units` <"pixels" | "common" | "meters"> The units of the size specified by `get_size`.

`point_size` <number> The radius of all points in units specified by `size_units`.

`get_position` <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_normal` <accessor | numeric> The normal of each object, in `c(nx, ny, nz)`. Accepts a length-3 numeric vector, or a tidy-eval 3-column matrix column.

`get_color` <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`material` <boolean>

`blending_mode` <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- `normal`: Normal blending doesn't alter colours of overlapping objects.
- `additive`: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- `subtractive`: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

`visibility_toggle`
<boolean> Whether this layer will appear in the layer selector.

`tooltip` <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if `pickable == TRUE`. Supports tidy-select if a `data` is a `data.frame`. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/point-cloud-layer.md

Other core-layers: `arc_layer`, `bitmap_layer`, `column_layer`, `geojson_layer`, `grid_cell_layer`, `icon_layer`, `line_layer`, `path_layer`, `polygon_layer`, `scatterplot_layer`, `solid_polygon_layer`, `text_layer`

Other layers: `arc_layer`, `bitmap_layer`, `column_layer`, `contour_layer`, `cpu_grid_layer`, `geojson_layer`, `gpu_grid_layer`, `great_circle_layer`, `grid_cell_layer`, `grid_layer`, `h3_cluster_layer`, `h3_hexagon_layer`, `heatmap_layer`, `hexagon_layer`, `icon_layer`, `line_layer`, `mvt_layer`, `path_layer`, `polygon_layer`, `quadkey_layer`, `s2_layer`, `scatterplot_layer`, `scenegraph_layer`, `screen_grid_layer`, `simple_mesh_layer`, `solid_polygon_layer`, `terrain_layer`, `text_layer`, `tile_3d_layer`, `tile_layer`, `trips_layer`

polygon_layer *Polygon Layer*

Description

Polygon Layer

Usage

```
add_polygon_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "PolygonLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  stroked = TRUE,
  filled = TRUE,
  extruded = FALSE,
  elevation_scale = 1,
  wireframe = FALSE,
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  line_joint_rounded = FALSE,
  line_miter_limit = 4,
  get_polygon = polygon,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_polygon_layer(
```

```
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  stroked = cur_value(),
  filled = cur_value(),
  extruded = cur_value(),
  elevation_scale = cur_value(),
  wireframe = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  line_joint_rounded = cur_value(),
  line_miter_limit = cur_value(),
  get_polygon = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

group_name <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

data <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

visible <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

stroked <boolean> If TRUE, draw an outline around each object.

filled <boolean> If TRUE, draw the filled area of each point.

extruded <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

elevation_scale
<number> The elevation multiplier.

wireframe <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

line_width_units
<"pixels" | "common" | "meters"> The units of line width.

line_width_scale
<number> The line width multiplier.

line_width_min_pixels
<number> The minimum line width in pixels.

line_width_max_pixels
<number> The maximum line width in pixels.

line_joint_rounded
<boolean>

line_miter_limit
number

get_polygon <accessor> The feature polygons. A <polygon/multipolygon> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_fill_color <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_color <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_width <accessor | scale | number> The outline of the object in units specified by line_width_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_elevation <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

material <boolean>

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/polygon-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, scatterplot_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

power_trans *Power transformation*

Description

Applies an exponential transform of the input: `y = abs(x)^k`, where:

- `x` is the input vector
- `k` is the exponent

Usage

```
power_trans(exponent = 0.5)
```

Arguments

`exponent`	<number> The power exponent

Details

If `x` is negative, the result is multiplied by -1.

See Also

Other transform: `log_trans()`, `symlog_trans()`

props *Props*

Description

Get map props

Usage

```
props(rdeck)
```

Arguments

`rdeck`	an rdeck instance

quadkey_layer *Quadkey Layer*

Description

Quadkey Layer

Usage

```
add_quadkey_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "QuadkeyLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  stroked = TRUE,
  filled = TRUE,
  extruded = FALSE,
  elevation_scale = 1,
  wireframe = FALSE,
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  line_joint_rounded = FALSE,
  line_miter_limit = 4,
  get_polygon = polygon,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  get_quadkey = quadkey,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)
```

```
update_quadkey_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  stroked = cur_value(),
  filled = cur_value(),
  extruded = cur_value(),
  elevation_scale = cur_value(),
  wireframe = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  line_joint_rounded = cur_value(),
  line_miter_limit = cur_value(),
  get_polygon = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  get_quadkey = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.

name <string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

group_name <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

data <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

visible <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

stroked <boolean> If TRUE, draw an outline around each object.

filled <boolean> If TRUE, draw the filled area of each point.

extruded <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

elevation_scale
<number> The elevation multiplier.

wireframe <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

line_width_units
<"pixels" | "common" | "meters"> The units of line width.

line_width_scale
<number> The line width multiplier.

line_width_min_pixels
<number> The minimum line width in pixels.

line_width_max_pixels
<number> The maximum line width in pixels.

line_joint_rounded
<boolean>

line_miter_limit
: number

get_polygon
: <accessor> The feature polygons. A <polygon/multipolygon> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_fill_color
: <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_color
: <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_width
: <accessor | scale | number> The outline of the object in units specified by line_width_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_elevation
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

material
: <boolean>

get_quadkey
: <accessor> The column containing the quadkey identifier.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/quadkey-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, mvt_layer, s2_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

rdeck *RDeck*

Description

Create a Deck.GL map. Rendering the mapbox basemap requires a mapbox account and mapbox access token.

Usage

```
rdeck(
  map_style = mapbox_dark(),
  theme = "kepler",
  initial_bounds = NULL,
  initial_view_state = view_state(center = c(0, 0), zoom = 1),
  controller = TRUE,
  picking_radius = 0,
  use_device_pixels = TRUE,
  blending_mode = "normal",
  layer_selector = TRUE,
  editor = FALSE,
  lazy_load = deprecated(),
  width = NULL,
  height = NULL,
  id = NULL,
  ...
)
```

Arguments

map_style	<string> The mapbox basemap style url. See https://docs.mapbox.com/api/maps/#mapbox-styles
theme	<"kepler" | "light"> The widget theme which alters the style of the legend and tooltips.
initial_bounds	<rct/st_bbox/wk-geometry> Sets the initial bounds of the map if not NULL. Takes priority over initial_view_state. Accepts a bounding box, or a geometry from which a bounding box can be computed. Requires CRS EPSG:4326.
initial_view_state	<view_state> Defines the map position, zoom, bearing and pitch.
controller	<logical> If NULL or FALSE, the map is not interactive.
picking_radius	<number> Extra pixels around the pointer to include while picking; useful when rendering objects that are difficult to hover, e.g. thin lines, small points, etc.
use_device_pixels	<logical | number> Controls the resolution of drawing buffer used for rendering.

- TRUE: Resolution is defined by window.devicePixelRatio. On Retina/HD displays, this resolution is usually twice as big as CSS pixels resolution.
- FALSE: CSS pixels resolution is used for rendering.
- number: Custom ratio (drawing buffer resolution to CSS pixel) to determine drawing buffer size. A value less than 1 uses resolution smaller than CSS pixels, improving rendering performance at the expense of image quality; a value greater than 1 improves image quality at the expense of rendering performance.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

layer_selector <boolean> If TRUE, the layer selector control will be enabled and layers with visibility_toggle = TRUE may be toggled. If FALSE, the layer selector control won't be rendered.

editor <booleaneditor_options> Whether to render the polygon editor. If TRUE, renders with the default editor_options(). If FALSE, the polygon editor is not rendered.

lazy_load **[Deprecated]**. Maps are always eagerly rendered.

width <number> The width of the map canvas.

height <number> The height of the map canvas.

id <string> The map element id. Not used in shiny applications.

... Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/core/deck.md

---

rdeck-shiny *Shiny bindings for rdeck*

Description

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

Usage

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

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

Arguments

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

---

rdeck_proxy *RDeck proxy*

Description

Creates an rdeck() interface for asynchronous updates of a pre-rendered rdeck map in Shiny apps.

All rdeck props can be updated through the proxy (NULL values will be discarded), layers that are added to the proxy (e.g. rdeck_proxy %>% add_h3_hexagon_layer()) will be merged with pre-rendered rdeck layers.

Layers are merged by their id. Matched layers will be updated in place, new layers will be appended and hence drawn *on top* of all existing layers. For layer updates, you may omit the data prop to avoid re-serialising unchanged data. All other props will assume their defaults if omitted.

Usage

```
rdeck_proxy(
  id,
  session = shiny::getDefaultReactiveDomain(),
  map_style = cur_value(),
  theme = cur_value(),
  initial_bounds = cur_value(),
  initial_view_state = cur_value(),
  controller = cur_value(),
  picking_radius = cur_value(),
  use_device_pixels = cur_value(),
  blending_mode = cur_value(),
  layer_selector = cur_value(),
  editor = cur_value(),
  lazy_load = deprecated(),
  ...
)
```

Arguments

id	<string> The map id
session	<ShinySession> The shiny session
map_style	<string> The mapbox basemap style url. See https://docs.mapbox.com/api/maps/#mapbox-styles
theme	<"kepler" | "light"> The widget theme which alters the style of the legend and tooltips.
initial_bounds	<rct/st_bbox/wk-geometry> Sets the initial bounds of the map if not NULL. Takes priority over initial_view_state. Accepts a bounding box, or a geometry from which a bounding box can be computed. Requires CRS EPSG:4326.
initial_view_state	<view_state> Defines the map position, zoom, bearing and pitch.
controller	<logical> If NULL or FALSE, the map is not interactive.
picking_radius	<number> Extra pixels around the pointer to include while picking; useful when rendering objects that are difficult to hover, e.g. thin lines, small points, etc.
use_device_pixels	<logical | number> Controls the resolution of drawing buffer used for rendering. • TRUE: Resolution is defined by window.devicePixelRatio. On Retina/HD displays, this resolution is usually twice as big as CSS pixels resolution. • FALSE: CSS pixels resolution is used for rendering. • number: Custom ratio (drawing buffer resolution to CSS pixel) to determine drawing buffer size. A value less than 1 uses resolution smaller than CSS pixels, improving rendering performance at the expense of image quality; a value greater than 1 improves image quality at the expense of rendering performance.
blending_mode	<"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes: • normal: Normal blending doesn't alter colours of overlapping objects. • additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps. • subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.
layer_selector	<boolean> If TRUE, the layer selector control will be enabled and layers with visibility_toggle = TRUE may be toggled. If FALSE, the layer selector control won't be rendered.
editor	<booleaneditor_options> Whether to render the polygon editor. If TRUE, renders with the default editor_options(). If FALSE, the polygon editor is not rendered.
lazy_load	**[Deprecated]**. Maps are always eagerly rendered.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.

Examples

```
## Not run:
library(shiny)
library(dplyr)
library(h3jsr)
library(viridis)

ui <- fillPage(
  rdeckOutput("map", height = "100%"),
  absolutePanel(
    top = 10, left = 10,
    sliderInput("range", "value", 0, 1, c(0, 1), step = 0.1)
  )
)

h3_data <- tibble(
  hexagon = get_res0() %>%
    get_children(res = 3) %>%
    unlist() %>%
    unique(),
  value = runif(length(hexagon))
)

map <- rdeck() %>%
  add_h3_hexagon_layer(
    id = "h3_hexagon",
    name = "hexagons",
    data = h3_data,
    get_fill_color = scale_color_quantize(
      col = value,
      palette = viridis(6, 0.3)
    ),
    pickable = TRUE,
    auto_highlight = TRUE,
    tooltip = c(hexagon, value)
  )

server <- function(input, output, session) {
  output$map <- renderRdeck(map)

  filtered_data <- reactive({
    h3_data %>%
      filter(value >= input$range[1] & value <= input$range[2])
  })

  observe({
    rdeck_proxy("map") %>%
      add_h3_hexagon_layer(
        id = "h3_hexagon",
        name = "hexagons",
        data = filtered_data(),
        get_fill_color = scale_color_quantize(
```

```
            col = value,
            palette = cividis(6, 0.3)
          ),
          pickable = TRUE,
          auto_highlight = TRUE,
          tooltip = c(hexagon, value)
        )
    })
}

app <- shinyApp(ui, server)

## End(Not run)
```

`rescale_center` *Rescale center*

Description

Re-centres a scale to have a defined centre / midpoint. This is the rdeck equivalent of `scales::rescale_mid()`.

centring an rdeck scale creates a new scale with the output palette or range centred at `center`. This is similar to creating a diverging scale; the key difference is that the output palette or range remains linear (with respect to the breaks) and is truncated on the side that is closest to `center`. This is useful in creating *difference* layers, where the output palette or range represents distance from the centre.

Usage

```
rescale_center(scale, center = 0)
```

Arguments

`scale`	<scale> a scale object
`center`	<number> the center of the scale input

Centring vs Diverging

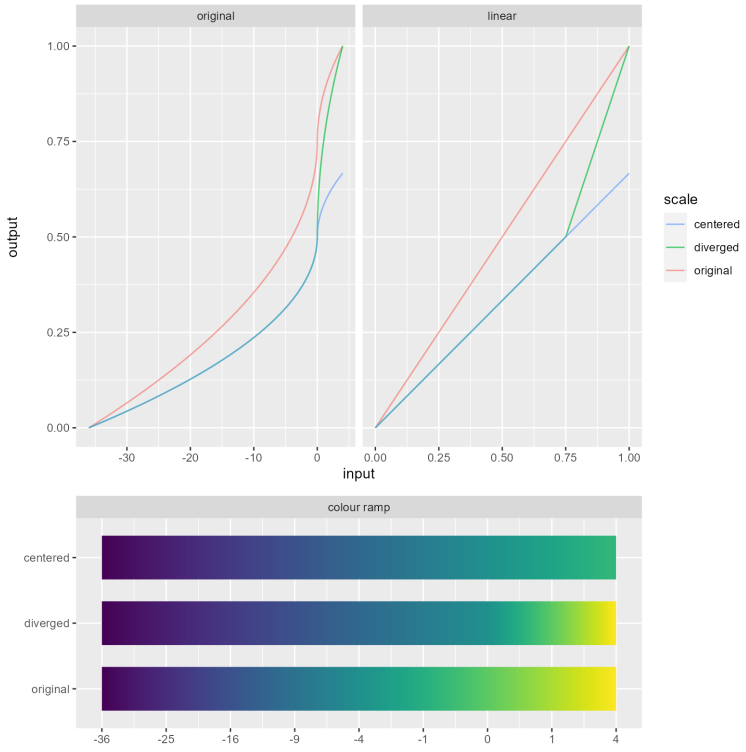
The plot below shows how `rescale_center()` and `rescale_diverge()` distort the scale output. The input scale in this case is `power_scale(limits = -36:4)`; this scale is centred and diverged at 0.

The plot on the left shows the mapping between the input `-36:4` (x axis) and output `0:1` (y axis). The plot on the right is a linear representation of the left and is the space that rdeck works in. The input `-36:4` transformed with `power_trans()` and rescaled to `0:1`. This plot has been included because it's (hopefully) easier to understand.

In the unaltered scale, we see that 0 is mapped to 0.75 in the output, which would be the colour at 0.75 on a colour ramp (e.g. `scales::colour_ramp(viridis::viridis(256))(0.75)`).

When applying `rescale_center()` we see that gradient of function has become y = 2/3x in the linear scale, which is `2/3 * scales::rescale(trans$transform(x))` for our data. For `rescale_diverge()` we see a piecewise scale with the break at `center`; both sides of `center` have a different gradient (y = 2/3x and y = 2x - 1) and the full range of y is used.

The colour ramp plot shows the effect rescaling has on a colour palette (in this case viridis).



Note

Category and identity scales aren't supported.

See Also

Other scales: `rescale_diverge()`, `scale_category()`, `scale_identity()`, `scale_linear()`, `scale_log()`, `scale_power()`, `scale_quantile()`, `scale_quantize()`, `scale_symlog()`, `scale_threshold()`

Examples

```
# create a sqrt scale that is centered at 0
sqrt_centered <- rescale_center(
  scale_color_power(col, limits = -36:4),
  center = 0
)

# create a discrete symlog scale that is centered at 5
symlog_centered <- rescale_center(
```

```
  scale_color_threshold(col, limits = -100:100, breaks = breaks_symlog()),
  center = 5
)
```

`rescale_diverge` *Rescale diverge*

Description

Creates a diverging scale with defined centre / midpoint. Similar to `rescale_center()`, key difference is the output palette / range is piecewise linear (with respect to breaks) and the entire output range is always used.

Usage

```
rescale_diverge(scale, center = 0)
```

Arguments

`scale`	<scale> a scale object
`center`	<number> the center of the scale input

Centring vs Diverging

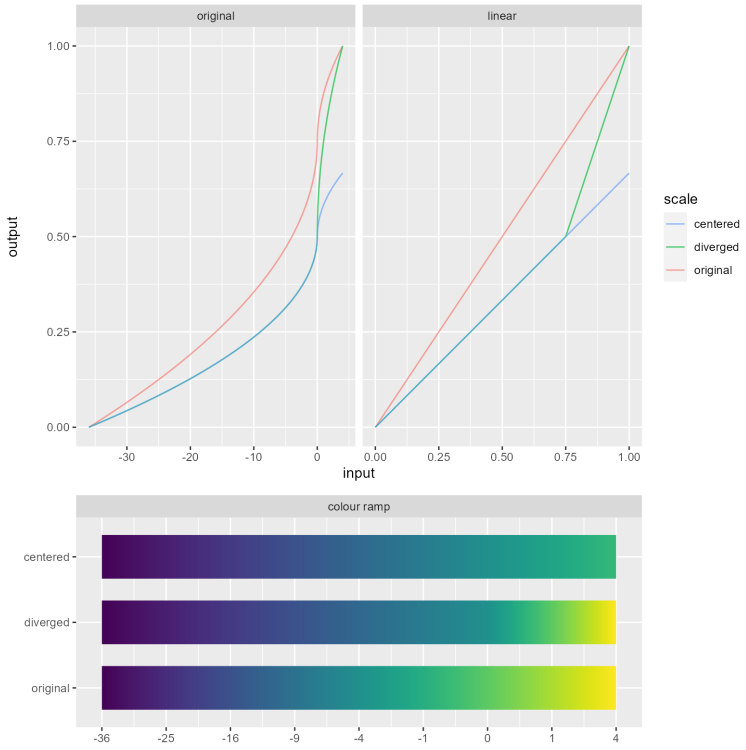
The plot below shows how `rescale_center()` and `rescale_diverge()` distort the scale output. The input scale in this case is `power_scale(limits = -36:4)`; this scale is centred and diverged at 0.

The plot on the left shows the mapping between the input `-36:4` (x axis) and output `0:1` (y axis). The plot on the right is a linear representation of the left and is the space that rdeck works in. The input `-36:4` transformed with `power_trans()` and rescaled to `0:1`. This plot has been included because it's (hopefully) easier to understand.

In the unaltered scale, we see that 0 is mapped to 0.75 in the output, which would be the colour at 0.75 on a colour ramp (e.g. `scales::colour_ramp(viridis::viridis(256))(0.75)`).

When applying `rescale_center()` we see that gradient of function has become y = 2/3x in the linear scale, which is `2/3 * scales::rescale(trans$transform(x))` for our data. For `rescale_diverge()` we see a piecewise scale with the break at `center`; both sides of `center` have a different gradient (y = 2/3x and y = 2x - 1) and the full range of y is used.

The colour ramp plot shows the effect rescaling has on a colour palette (in this case viridis).



Note

Category and identity scales aren't supported.

See Also

Other scales: `rescale_center()`, `scale_category()`, `scale_identity()`, `scale_linear()`, `scale_log()`, `scale_power()`, `scale_quantile()`, `scale_quantize()`, `scale_symlog()`, `scale_threshold()`

Examples

```
# create a diverging linear scale at 0
linear_diverged <- rescale_diverge(
  scale_color_linear(col, limits = -5:10),
  center = 0
)

# create a diverging log scale at 10
log_diverged <- rescale_diverge(
  scale_log(col, limits = 1:1000),
  center = 10
)
```

s2_layer *S2 Layer*

Description

S2 Layer

Usage

```
add_s2_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "S2Layer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  stroked = TRUE,
  filled = TRUE,
  extruded = FALSE,
  elevation_scale = 1,
  wireframe = FALSE,
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  line_joint_rounded = FALSE,
  line_miter_limit = 4,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  get_elevation = 1000,
  material = TRUE,
  get_s2_token = token,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_s2_layer(
```

```
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  stroked = cur_value(),
  filled = cur_value(),
  extruded = cur_value(),
  elevation_scale = cur_value(),
  wireframe = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  line_joint_rounded = cur_value(),
  line_miter_limit = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  get_elevation = cur_value(),
  material = cur_value(),
  get_s2_token = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

group_name <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

data <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

visible <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

stroked <boolean> If TRUE, draw an outline around each object.

filled <boolean> If TRUE, draw the filled area of each point.

extruded <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

elevation_scale
<number> The elevation multiplier.

wireframe <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

line_width_units
<"pixels" | "common" | "meters"> The units of line width.

line_width_scale
<number> The line width multiplier.

line_width_min_pixels
<number> The minimum line width in pixels.

line_width_max_pixels
<number> The maximum line width in pixels.

line_joint_rounded
<boolean>

line_miter_limit
number

get_fill_color <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_color <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_width <accessor | scale | number> The outline of the object in units specified by line_width_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_elevation <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

material <boolean>

get_s2_token <accessor> The S2 hex token for each S2 cell. Accepts a tidy-eval character column.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/s2-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

scale_category *Scale category*

Description

Creates a categorical scale. Categorical scales map input values defined in the set of `levels` to colours (or values). Input values not in the set of `levels` are assigned `unmapped_color` (or `unmapped_value`).

Usage

```
scale_color_category(
  col,
  palette = scales::brewer_pal("div"),
  unmapped_color = "#000000",
  levels = NULL,
  unmapped_tick = NULL,
  tick_format = NULL,
  col_label = "{.col}",
  legend = TRUE
)

scale_category(
  col,
  range = 0:1,
  unmapped_value = 0,
  levels = NULL,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

`col`	<`name` | `string`> The name of the column containing data to be scaled. Must be a valid input to `rlang::ensym()`; either a named column (non-standard evaluation), a string. Supports tidy-eval.
`palette`	<`color`|`function`> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from `scales::colour_ramp()` A `scales::colour_ramp()` interpolator is created from the input palette.
`unmapped_color`	<`color`> The colour representing unmapped levels.
`levels`	<`factor` | `character` | `logical`> The category levels. If `NULL`, will be populated from input data. The order of the levels is determined by `levels()` for factors & `unique()` otherwise.

	If there are more levels than colours (or range values), the palette (or range) is interpolated.
unmapped_tick	<string> The tick label of the unmapped category. If not NULL and legend == TRUE, the unmapped category will appear at the bottom of the legend.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
unmapped_value	<number> The value representing unmapped levels.

See Also

Other scales: rescale_center(), rescale_diverge(), scale_identity(), scale_linear(), scale_log(), scale_power(), scale_quantile(), scale_quantize(), scale_symlog(), scale_threshold()

---

scale_identity *Scale identity*

Description

Creates an identity scale; a special case of a linear scale, where input is mapped to itself (input limits = output range). An identity scale is useful in cases where input data is already expressed in a visual representation (e.g. a line width) and should be used as-is.

Usage

```
scale_identity(col, na_value = 0, col_label = "{.col}", legend = TRUE)
```

Arguments

col	<name | string> The name of the column containing data to be scaled. Must be a valid input to rlang::ensym(); either a named column (non-standard evaluation), a string. Supports tidy-eval.
na_value	<number> The output value for NA input values.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.

Note

Identity scales are *almost* equivalent to an accessor to a numeric column; differences are:

- NA is replaced with na_value
- May render a *numeric* legend

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_linear(), scale_log(), scale_power(), scale_quantile(), scale_quantize(), scale_symlog(), scale_threshold()

---

scale_linear *Scale linear*

Description

Creates a continuous linear scale. The colour palette (or range) is linearly interpolated between limits (or between limits and between breaks for piecewise scales).

Usage

```
scale_color_linear(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  limits = NULL,
  breaks = NULL,
  n_ticks = NULL,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_linear(
  col,
  range = 0:1,
  na_value = 0,
  limits = NULL,
  breaks = NULL,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

col	<name | string> The name of the column containing data to be scaled. Must be a valid input to rlang::ensym(); either a named column (non-standard evaluation), a string. Supports tidy-eval.
palette	<color|function> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from scales::colour_ramp() A scales::colour_ramp() interpolator is created from the input palette.
na_color	<color> The colour value for NA input values.
limits	<c(min, max)> The limits of the scale's input. If NULL, limits are computed from layer data. Values outside the range of limits are clamped.
breaks	<numeric | function> The breaks of the scale, allowing to define a piecewise scale. The scale palette or numeric range are linearly interpolated (by length) and mapped onto breaks. Breaks outside the limits of the scale are discarded. If not NULL, breaks must be either: • a numeric vector • a breaks function, taking a numeric vector argument (e.g. breaks_linear()) Defaults to breaks_trans() where trans is the scale's transformer.
n_ticks	<number> The number of ticks to display on the legend. Must be >= 2. The legend size will grow and shrink depending on this value.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
na_value	<number> The output value for NA input values.

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_identity(), scale_log(), scale_power(), scale_quantile(), scale_quantize(), scale_symlog(), scale_threshold()

scale_log *Scale log*

Description

Creates a continuous log scale, where input values are transformed with log_trans(base) before calculating the output.

Log scales can be useful in transforming positively skewed data.

Usage

```
scale_color_log(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  base = 10,
  limits = NULL,
  breaks = NULL,
  n_ticks = NULL,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_log(
  col,
  range = 0:1,
  na_value = 0,
  base = 10,
  limits = NULL,
  breaks = NULL,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

col	<name | string> The name of the column containing data to be scaled. Must be a valid input to rlang::ensym(); either a named column (non-standard evaluation), a string. Supports tidy-eval.
palette	<color|function> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from scales::colour_ramp() A scales::colour_ramp() interpolator is created from the input palette.

na_color	<color> The colour value for NA input values.
base	<number> The log base. The log base must be a strictly positive value != 1.
limits	<c(min, max)> The limits of the scale's input. If NULL, limits are computed from layer data. Values outside the range of limits are clamped.
breaks	<numeric | function> The breaks of the scale, allowing to define a piecewise scale. The scale palette or numeric range are linearly interpolated (by length) and mapped onto breaks. Breaks outside the limits of the scale are discarded. If not NULL, breaks must be either: • a numeric vector • a breaks function, taking a numeric vector argument (e.g. breaks_linear()) Defaults to breaks_trans() where trans is the scale's transformer.
n_ticks	<number> The number of ticks to display on the legend. Must be >= 2. The legend size will grow and shrink depending on this value.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
na_value	<number> The output value for NA input values.

Note

limits, whether explicitly supplied, or computed from data, must not cross 0.

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_identity(), scale_linear(), scale_power(), scale_quantile(), scale_quantize(), scale_symlog(), scale_threshold()

---

scale_power *Scale power*

Description

Creates a continuous power scale, where input values are transformed with power_trans(exponent) before calculating the output.

Power scales can be useful in transforming positively skewed data. A square-root or cube-root scale can be helpful in dealing with right-skewed data.

A square-root scale can be defined with scale_power(exponent = 0.5, ...) (the default). A square-root scale is a good choice for scaling the radius of point data, as this would result in a linear scale for the area of each point.

Usage

```
scale_color_power(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  exponent = 0.5,
  limits = NULL,
  breaks = NULL,
  n_ticks = 6,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_power(
  col,
  range = 0:1,
  na_value = 0,
  exponent = 0.5,
  limits = NULL,
  breaks = NULL,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

col	<name | string> The name of the column containing data to be scaled. Must be a valid input to rlang::ensym(); either a named column (non-standard evaluation), a string. Supports tidy-eval.
palette	<color|function> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from scales::colour_ramp() A scales::colour_ramp() interpolator is created from the input palette.
na_color	<color> The colour value for NA input values.
exponent	<number> The power exponent.
limits	<c(min, max)> The limits of the scale's input. If NULL, limits are computed from layer data. Values outside the range of limits are clamped.
breaks	<numeric | function> The breaks of the scale, allowing to define a piecewise scale. The scale palette or numeric range are linearly interpolated (by length) and mapped onto breaks. Breaks outside the limits of the scale are discarded. If not NULL, breaks must be either: • a numeric vector • a breaks function, taking a numeric vector argument (e.g. breaks_linear())

	Defaults to `breaks_trans()` where `trans` is the scale's transformer.
`n_ticks`	<number> The number of ticks to display on the legend. Must be >= 2. The legend size will grow and shrink depending on this value.
`tick_format`	<function> A label function taking a vector of ticks returning formatted ticks.
`col_label`	<string | function> A template string or a label function for customising the column name in the legend. • if `col_label` is a string, `{.col}` may be used to represent the `col` name • if `col_label` is a function, the function must take a single argument: the `col` name
`legend`	<boolean> Indicate whether the legend should be displayed for this scale.
`range`	<numeric> The output range of the numeric scale. Must be length >= 2. A `stats::approxfun()` interpolator is created from the input range.
`na_value`	<number> The output value for `NA` input values.

See Also

Other scales: `rescale_center()`, `rescale_diverge()`, `scale_category()`, `scale_identity()`, `scale_linear()`, `scale_log()`, `scale_quantile()`, `scale_quantize()`, `scale_symlog()`, `scale_threshold()`

---

scale_quantile *Scale quantile*

Description

Creates a threshold scale, where threshold breaks are computed from the given quantile `probs`.

Quantile scale legend ticks will be quantile values at each quantile break (including limits), not the quantile probabilities at each break. This can be overridden in `tick_format`.

Example: `tick_format = function(x) as.character(probs)`.

Usage

```
scale_color_quantile(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  probs = seq.int(0, 1, 0.25),
  data = NULL,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_quantile(
  col,
```

```
  range = 0:1,
  na_value = 0,
  probs = seq.int(0, 1, 0.25),
  data = NULL,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

col	<name | string> The name of the column containing data to be scaled. Must be a valid input to rlang::ensym(); either a named column (non-standard evaluation), a string. Supports tidy-eval.
palette	<color|function> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from scales::colour_ramp() A scales::colour_ramp() interpolator is created from the input palette.
na_color	<color> The colour value for NA input values.
probs	<numeric> The quantile probabilities. Must be between 0 and 1.
data	<numeric> The data used to compute the quantiles. If NULL, will be taken from the layer data.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
na_value	<number> The output value for NA input values.

Note

As the quantiles are computed from input data, quantile scales are incompatible with layers that load data from a url (e.g mvt_layer). If quantiles for remote data are known, a quantile scale can be constructed manually with scale_threshold().

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_identity(), scale_linear(), scale_log(), scale_power(), scale_quantize(), scale_symlog(), scale_threshold()

scale_quantize *Scale quantize*

Description

Creates a discrete quantize scale, with `n_breaks` linearly spaced threshold breaks between `limits`. This scale can be thought of as a restricted special case of `scale_threshold()`, or a discrete `scale_linear()`.

Usage

```
scale_color_quantize(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  limits = NULL,
  n_breaks = 6,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_quantize(
  col,
  range = 0:1,
  na_value = 0,
  limits = NULL,
  n_breaks = 6,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

`col`	<name | `string`> The name of the column containing data to be scaled. Must be a valid input to `rlang::ensym()`; either a named column (non-standard evaluation), a string. Supports tidy-eval.
`palette`	<`color`|`function`> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from `scales::colour_ramp()` A `scales::colour_ramp()` interpolator is created from the input palette.
`na_color`	<`color`> The colour value for `NA` input values.
`limits`	<`c(min, max)`> The limits of the scale's input. If `NULL`, limits are computed from layer data. Values outside the range of limits are clamped.

n_breaks	<integer> The number of linearly spaced breaks in the scale. Each break will be present on the legend for colour scales.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
na_value	<number> The output value for NA input values.

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_identity(), scale_linear(), scale_log(), scale_power(), scale_quantile(), scale_symlog(), scale_threshold()

---

scale_symlog *Scale symlog*

Description

Creates a continuous log1p scale, where input values are transformed with symlog_trans() before calculating the output. Unlike scale_log(), limits may cross 0.

Symlog scales can be useful in transforming positively skewed data.

Usage

```
scale_color_symlog(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  limits = NULL,
  breaks = NULL,
  n_ticks = NULL,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_symlog(
  col,
  range = 0:1,
  na_value = 0,
```

```
  limits = NULL,
  breaks = NULL,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

col	<name | string> The name of the column containing data to be scaled. Must be a valid input to rlang::ensym(); either a named column (non-standard evaluation), a string. Supports tidy-eval.
palette	<color|function> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from scales::colour_ramp() A scales::colour_ramp() interpolator is created from the input palette.
na_color	<color> The colour value for NA input values.
limits	<c(min, max)> The limits of the scale's input. If NULL, limits are computed from layer data. Values outside the range of limits are clamped.
breaks	<numeric | function> The breaks of the scale, allowing to define a piecewise scale. The scale palette or numeric range are linearly interpolated (by length) and mapped onto breaks. Breaks outside the limits of the scale are discarded. If not NULL, breaks must be either: • a numeric vector • a breaks function, taking a numeric vector argument (e.g. breaks_linear()) Defaults to breaks_trans() where trans is the scale's transformer.
n_ticks	<number> The number of ticks to display on the legend. Must be >= 2. The legend size will grow and shrink depending on this value.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
na_value	<number> The output value for NA input values.

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_identity(), scale_linear(), scale_log(), scale_power(), scale_quantile(), scale_quantize(), scale_threshold()

scale_threshold *Scale threshold*

Description

Creates a discrete threshold scale. Threshold scales slice `palette` or `range` into `length(breaks) + 1` bins, with each break defining the threshold between 2 bins.

Threshold scales can be used to create any discrete scale, using either manual breaks or generated breaks via a transform (e.g. breaks_power(n = 6, exponent = 0.5) for a discrete sqrt scale).

Usage

```
scale_color_threshold(
  col,
  palette = scales::viridis_pal(),
  na_color = "#000000",
  limits = NULL,
  breaks = 0.5,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)

scale_threshold(
  col,
  range = 0:1,
  na_value = 0,
  limits = NULL,
  breaks = 0.5,
  tick_format = format_number,
  col_label = "{.col}",
  legend = TRUE
)
```

Arguments

`col`	<name | `string`> The name of the column containing data to be scaled. Must be a valid input to `rlang::ensym()`; either a named column (non-standard evaluation), a string. Supports tidy-eval.
`palette`	<`color`|`function`> The colour palette of the colour scale. Must be a: • vector of RGBA hex colours, • a palette generator function, taking a length parameter, or • a palette ramp created from `scales::colour_ramp()` A `scales::colour_ramp()` interpolator is created from the input palette.
`na_color`	<`color`> The colour value for `NA` input values.

limits	<c(min, max)> The limits of the scale's input. If NULL, limits are computed from layer data. Values outside the range of limits are clamped.
breaks	<numeric | function> The threshold breaks of the scale. The scale palette or numeric range are linearly interpolated (by length) and mapped onto breaks. Breaks outside the limits of the scale are discarded. If not NULL, breaks must be either: • a numeric vector • a breaks function, taking a numeric vector argument (e.g. breaks_linear()) Breaks must be in increasing order. Each break will be present on the legend for colour scales.
tick_format	<function> A label function taking a vector of ticks returning formatted ticks.
col_label	<string | function> A template string or a label function for customising the column name in the legend. • if col_label is a string, {.col} may be used to represent the col name • if col_label is a function, the function must take a single argument: the col name
legend	<boolean> Indicate whether the legend should be displayed for this scale.
range	<numeric> The output range of the numeric scale. Must be length >= 2. A stats::approxfun() interpolator is created from the input range.
na_value	<number> The output value for NA input values.

Note

Threshold scales don't require limits, but breaks_trans() does.

See Also

Other scales: rescale_center(), rescale_diverge(), scale_category(), scale_identity(), scale_linear(), scale_log(), scale_power(), scale_quantile(), scale_quantize(), scale_symlog()

---

scatterplot_layer *Scatterplot Layer*

Description

Scatterplot Layer

Usage

```
add_scatterplot_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "ScatterplotLayer",
```

```
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  radius_units = "meters",
  radius_scale = 1,
  radius_min_pixels = 0,
  radius_max_pixels = 9007199254740991,
  line_width_units = "meters",
  line_width_scale = 1,
  line_width_min_pixels = 0,
  line_width_max_pixels = 9007199254740991,
  stroked = FALSE,
  filled = TRUE,
  billboard = FALSE,
  antialiasing = TRUE,
  get_position = position,
  get_radius = 1,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  get_line_width = 1,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_scatterplot_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  radius_units = cur_value(),
```

```
  radius_scale = cur_value(),
  radius_min_pixels = cur_value(),
  radius_max_pixels = cur_value(),
  line_width_units = cur_value(),
  line_width_scale = cur_value(),
  line_width_min_pixels = cur_value(),
  line_width_max_pixels = cur_value(),
  stroked = cur_value(),
  filled = cur_value(),
  billboard = cur_value(),
  antialiasing = cur_value(),
  get_position = cur_value(),
  get_radius = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  get_line_width = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`
<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`radius_units` <"pixels" | "common" | "meters"> The units of point radius.

`radius_scale` <number> The radius multiplier for all points.

`radius_min_pixels`
<number> The minimum radius in pixels.

`radius_max_pixels`
<number> The maximum radius in pixels.

`line_width_units`
<"pixels" | "common" | "meters"> The units of line width.

`line_width_scale`
<number> The line width multiplier.

`line_width_min_pixels`
<number> The minimum line width in pixels.

`line_width_max_pixels`
<number> The maximum line width in pixels.

`stroked` <boolean> If TRUE, draw an outline around each object.

`filled` <boolean> If TRUE, draw the filled area of each point.

`billboard` <boolean> If TRUE, circles always face the camera; if FALSE circles face up (z).

`antialiasing` <boolean> If TRUE, circles are rendered with smoothed edges.

`get_position` <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

`get_radius` <accessor | scale | number> The radius of each point, in units specified by `radius_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`get_fill_color` <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_color` <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

`get_line_width` <accessor | scale | number> The outline of the object in units specified by `line_width_units`. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

`blending_mode` <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- `normal`: Normal blending doesn't alter colours of overlapping objects.

- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/scatterplot-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, solid_polygon_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

scenegraph_layer *Scenegraph Layer*

Description

Scenegraph Layer

Usage

```
add_scenegraph_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "ScenegraphLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
```

```
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  scenegraph = NULL,
  get_scene = NULL,
  get_animator = NULL,
  size_scale = 1,
  size_min_pixels = 0,
  size_max_pixels = 9007199254740991,
  get_position = position,
  get_color = "#ffffffff",
  get_orientation = c(0, 0, 0),
  get_scale = c(1, 1, 1),
  get_translation = c(0, 0, 0),
  get_transform_matrix = NULL,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_scenegraph_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  scenegraph = cur_value(),
  get_scene = cur_value(),
  get_animator = cur_value(),
  size_scale = cur_value(),
  size_min_pixels = cur_value(),
  size_max_pixels = cur_value(),
  get_position = cur_value(),
  get_color = cur_value(),
  get_orientation = cur_value(),
  get_scale = cur_value(),
  get_translation = cur_value(),
  get_transform_matrix = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
```

```
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.
highlight_color	<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.
scenegraph	object
get_scene	function
get_animator	function
size_scale	<number> The size multiplier.
size_min_pixels	<number> The minimum size in pixels.

size_max_pixels
: <number> The maximum size in pixels.

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_color
: <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_orientation
: <accessor> not yet supported.

get_scale
: <accessor> not yet supported.

get_translation
: <accessor> not yet supported.

get_transform_matrix
: <accessor> not yet supported.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/mesh-layers/scenegraph-layer.md

Other mesh-layers: simple_mesh_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

screen_grid_layer *Screen Grid Layer*

Description

Screen Grid Layer

Usage

```
add_screen_grid_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "ScreenGridLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  cell_size_pixels = 100,
  cell_margin_pixels = 2,
  color_domain = NULL,
 color_range = c("#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"),
  get_position = position,
  get_weight = 1,
  gpu_aggregation = TRUE,
  aggregation = "SUM",
  blending_mode = "normal",
  visibility_toggle = TRUE
)

update_screen_grid_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
```

```
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  cell_size_pixels = cur_value(),
  cell_margin_pixels = cur_value(),
  color_domain = cur_value(),
  color_range = cur_value(),
  get_position = cur_value(),
  get_weight = cur_value(),
  gpu_aggregation = cur_value(),
  aggregation = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
: <accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

cell_size_pixels
: <number> Unit width / height of the bins.

cell_margin_pixels
: <number> Cell margin size in pixels.

color_domain
: <number> The colour scale domain, default is set to the range of aggregated weights in each bin.

color_range
: <color> The colour palette. color_domain is divided into length(color_range) equal segments, each mapped to one color in color_range.

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_weight
: <accessor | scale | number> The weight of each object. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

gpu_aggregation
: <boolean> If TRUE, aggregation is performed on GPU if supported. Requires WebGL2.

aggregation
: <"SUM" | "MEAN" | "MIN" | "MAX"> Defines the aggregation function.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:
  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/aggregation-layers/screen-grid-layer.md

Other aggregation-layers: contour_layer, cpu_grid_layer, gpu_grid_layer, grid_layer, heatmap_layer, hexagon_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

set_layer_visibility *Set layer visibility*

Description

Sets a layer's visibility and whether it is *selectable* in the layer selector. Setting either `visible` or `visibility_toggle` as `cur_value()` will have no change in the browser.

Usage

```
set_layer_visibility(
  rdeck,
  id,
  visible = cur_value(),
  visibility_toggle = cur_value()
)
```

Arguments

`rdeck`	<rdeck_proxy> the rdeck proxy object
`id`	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to `uuid::UUIDgenerate()`, but should be explicitly defined for updatable layers in a shiny application.
`visible`	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
`visibility_toggle`	<boolean> Whether this layer will appear in the layer selector.

sf_column *SF Column*

Description

Defines a flag that indicates the active geometry column of an sf object should be used in a layer's geometry `accessor()`.

Usage

```
sf_column()
```

shiny-events *Shiny events data*

Description

Utilities for retrieving map event data in a shiny application.

Usage

```
get_view_bounds(rdeck, session = shiny::getDefaultReactiveDomain())

get_view_state(rdeck, session = shiny::getDefaultReactiveDomain())

get_clicked_coordinates(rdeck, session = shiny::getDefaultReactiveDomain())

get_clicked_layer(rdeck, session = shiny::getDefaultReactiveDomain())

get_clicked_object(rdeck, session = shiny::getDefaultReactiveDomain())

get_edited_features(rdeck, session = shiny::getDefaultReactiveDomain())
```

Arguments

rdeck	<rdeck_proxy | string> the map, or map id
session	<ShinySession> the shiny session

Functions

- get_view_bounds(): Get the current map bounding box
- get_view_state(): Get the map view state
- get_clicked_coordinates(): Get the last clicked coordinates
- get_clicked_layer(): Get the last clicked layer (or NULL)
- get_clicked_object(): Get the last clicked object (or NULL)
- get_edited_features(): Get the edited features

simple_mesh_layer *Simple Mesh Layer*

Description

Simple Mesh Layer

Usage

```
add_simple_mesh_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "SimpleMeshLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  mesh = NULL,
  texture = NULL,
  size_scale = 1,
  wireframe = FALSE,
  material = TRUE,
  get_position = position,
  get_color = "#000000ff",
  get_orientation = c(0, 0, 0),
  get_scale = c(1, 1, 1),
  get_translation = c(0, 0, 0),
  get_transform_matrix = NULL,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_simple_mesh_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  mesh = cur_value(),
```

```
  texture = cur_value(),
  size_scale = cur_value(),
  wireframe = cur_value(),
  material = cur_value(),
  get_position = cur_value(),
  get_color = cur_value(),
  get_orientation = cur_value(),
  get_scale = cur_value(),
  get_translation = cur_value(),
  get_transform_matrix = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.
position_format	<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.
color_format	<"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.
auto_highlight	<boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
: <accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

mesh
: object

texture
: not yet supported.

size_scale
: <number> The size multiplier.

wireframe
: <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

material
: <boolean>

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_color
: <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_orientation
: <accessor> not yet supported.

get_scale
: <accessor> not yet supported.

get_translation
: <accessor> not yet supported.

get_transform_matrix
: <accessor> not yet supported.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:
  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/mesh-layers/simple-mesh-layer.md

Other mesh-layers: scenegraph_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer,

h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

`solid_polygon_layer` *Solid Polygon Layer*

Description

Solid Polygon Layer

Usage

```
add_solid_polygon_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "SolidPolygonLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  filled = TRUE,
  extruded = FALSE,
  wireframe = FALSE,
  elevation_scale = 1,
  get_polygon = polygon,
  get_elevation = 1000,
  get_fill_color = "#000000ff",
  get_line_color = "#000000ff",
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_solid_polygon_layer(
  rdeck,
  ...,
  id,
```

```
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  filled = cur_value(),
  extruded = cur_value(),
  wireframe = cur_value(),
  elevation_scale = cur_value(),
  get_polygon = cur_value(),
  get_elevation = cur_value(),
  get_fill_color = cur_value(),
  get_line_color = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.

position_format
: <"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format
: <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight
: <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
: <accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

filled
: <boolean> If TRUE, draw the filled area of each point.

extruded
: <boolean> If TRUE, extrude objects along the z-axis; if FALSE, all objects will be flat.

wireframe
: <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

elevation_scale
: <number> The elevation multiplier.

get_polygon
: <accessor> The feature polygons. A <polygon/multipolygon> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_elevation
: <accessor | scale | number> The elevation to extrude each object in the z-axis. Height units are in metres. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_fill_color
: <accessor | scale | color> The fill colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_line_color
: <accessor | scale | color> The line colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

material
: <boolean>

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
  - subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/solid-polygon-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, text_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

symlog_trans *SymLog transformation*

Description

Applies a log1p transform: `y = log1p(abs(x))`, where:

- `x` is the input vector

Usage

```
symlog_trans()
```

Details

If `x` is negative, the result is multiplied by -1.

See Also

Other transform: `log_trans()`, `power_trans()`

---

terrain_layer *Terrain Layer*

Description

Terrain Layer

Usage

```
add_terrain_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "TerrainLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_tile_data = NULL,
  extent = NULL,
  tile_size = 512,
  max_zoom = NULL,
  min_zoom = 0,
  max_cache_size = NULL,
  max_cache_byte_size = NULL,
  refinement_strategy = "best-available",
  z_range = NULL,
  max_requests = 6,
  zoom_offset = 0,
  elevation_data = NULL,
  texture = NULL,
  mesh_max_error = 4,
  bounds = NULL,
  color = "#ffffff",
  elevation_decoder = list(rScaler = 1, gScaler = 0, bScaler = 0, offset = 0),
  worker_url = NULL,
  wireframe = FALSE,
  material = TRUE,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_terrain_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
```

```
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_tile_data = cur_value(),
  extent = cur_value(),
  tile_size = cur_value(),
  max_zoom = cur_value(),
  min_zoom = cur_value(),
  max_cache_size = cur_value(),
  max_cache_byte_size = cur_value(),
  refinement_strategy = cur_value(),
  z_range = cur_value(),
  max_requests = cur_value(),
  zoom_offset = cur_value(),
  elevation_data = cur_value(),
  texture = cur_value(),
  mesh_max_error = cur_value(),
  bounds = cur_value(),
  color = cur_value(),
  elevation_decoder = cur_value(),
  worker_url = cur_value(),
  wireframe = cur_value(),
  material = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will

contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

`visible` <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

`pickable` <boolean> Determines if the layer responds to pointer / touch events.

`opacity` <number> Determines the layer's opacity.

`wrap_longitude` <boolean> Normalises geometry longitudes.

`position_format`
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`
<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`get_tile_data` <JS> retrieves the data of each tile. See deck.gl getTileData.

`extent` <c(min_x, min_y, max_x, max_y)> Tiles in this bounding box will be rendered at `min_zoom`, when zoomed out below `min_zoom`.

`tile_size` <number> A power of 2 that is the pixel dimensions of the tile.

`max_zoom` <number> Tiles above this zoom level are not shown. Defaults to `NULL`.

`min_zoom` <number> Tiles below this zoom level are not shown. Defaults to `0`.

`max_cache_size` <number> Maximum number of tiles that can be cached. Defaults to 5x the number of tiles in current viewport.

`max_cache_byte_size`
<number> Maximum memory used for caching tiles.

`refinement_strategy`
<"best-available" | "no-overlap" | "never"> How the tile layer refines visibility of tiles. Defaults to "best-available".

`z_range` <c(min, max)> Array representing the range of heights in the tile.

`max_requests` <number> Maximum number of concurrent HTTP requests across all specified tile provider domains. If a negative number is supplied no throttling occurs (HTTP/2 only).

`zoom_offset` <int> The offset changes the zoom level at which the tiles are fetched.

`elevation_data` url

`texture` not yet supported.

`mesh_max_error` number

bounds <rct/st_bbox/wk-geometry> The bounds of the image to fit x,y coordinates into. Requires CRS EPSG:4326. Must be supplied when using non-tiled elevation data.

color color

elevation_decoder
object

worker_url string

wireframe <boolean> If TRUE and extruded == TRUE, draw a line wireframe of the object. The outline will have horizontal lines closing the top and bottom polygons and vertical lines for each vertex of the polygon.

material <boolean>

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/terrain-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, s2_layer, tile_3d_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, text_layer, tile_3d_layer, tile_layer, trips_layer

---

text_layer *Text Layer*

Description

Text Layer

Usage

```
add_text_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "TextLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  billboard = TRUE,
  size_scale = 1,
  size_units = "pixels",
  size_min_pixels = 0,
  size_max_pixels = 9007199254740991,
  background = FALSE,
  get_background_color = "#ffffffff",
  get_border_color = "#000000ff",
  get_border_width = 0,
  background_padding = c(0, 0, 0, 0),
  font_family = "Roboto, Helvetica, Arial, san-serif",
  font_weight = "normal",
  line_height = 1,
  outline_width = 0,
  outline_color = "#000000ff",
  font_settings = list(),
  word_break = "break-word",
  max_width = -1,
  get_text = text,
  get_position = position,
  get_color = "#000000ff",
  get_size = 32,
  get_angle = 0,
  get_text_anchor = "middle",
  get_alignment_baseline = "center",
  get_pixel_offset = c(0, 0),
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_text_layer(
```

```
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  billboard = cur_value(),
  size_scale = cur_value(),
  size_units = cur_value(),
  size_min_pixels = cur_value(),
  size_max_pixels = cur_value(),
  background = cur_value(),
  get_background_color = cur_value(),
  get_border_color = cur_value(),
  get_border_width = cur_value(),
  background_padding = cur_value(),
  font_family = cur_value(),
  font_weight = cur_value(),
  line_height = cur_value(),
  outline_width = cur_value(),
  outline_color = cur_value(),
  font_settings = cur_value(),
  word_break = cur_value(),
  max_width = cur_value(),
  get_text = cur_value(),
  get_position = cur_value(),
  get_color = cur_value(),
  get_size = cur_value(),
  get_angle = cur_value(),
  get_text_anchor = cur_value(),
  get_alignment_baseline = cur_value(),
  get_pixel_offset = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.

`...` Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be `camelCased` when serialised. A warning is raised when dots are used, warning class `rdeck_dots_nonempty`.

`id` <`string`> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to `uuid::UUIDgenerate()`, but should be explicitly defined for updatable layers in a shiny application.

`name` <`string`> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

`group_name` <`string`> Defines the group that this layer belongs to. Currently only effective on the layer selector, if `visibility_toggle = TRUE`.

`data` <`data.frame` | `sf` | `string`> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

`visible` <`boolean`> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

`pickable` <`boolean`> Determines if the layer responds to pointer / touch events.

`opacity` <`number`> Determines the layer's opacity.

`wrap_longitude` <`boolean`> Normalises geometry longitudes.

`position_format` <`"XY" | "XYZ"`> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <`"RGB" | "RGBA"`> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <`boolean`> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color` <`accessor` | `scale` | `color`> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`billboard` <`boolean`> If `TRUE`, the text label always faces the camera, otherwise it faces up (z).

`size_scale` <`number`> The size multiplier.

`size_units` <`"pixels" | "common" | "meters"`> The units of the size specified by `get_size`.

`size_min_pixels` <`number`> The minimum size in pixels.

`size_max_pixels` <`number`> The maximum size in pixels.

`background` <`boolean`> Whether to render background for text labels.

`get_background_color` <`accessor` | `scale` | `color`> The text background colour, if `background = TRUE`. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_border_color
: <accessor | scale | color> The text background border colour, if background = TRUE. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_border_width
: <accessor | scale | number> The text background border width, if background = TRUE. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

background_padding
: <numeric> The text background padding. Must be an array of 2 or 4 numbers.

font_family
: <string> Specifies a prioritised list of one or more font family names. See font-family.

font_weight
: <"normal" | "bold" | 100:900> The font weight. See font-weight

line_height
: <number> A unitless number that will be multiplied with get_size to set the line height.

outline_width
: <number> The text outline width, relative to font size. Requires font_settings$sdf = TRUE.

outline_color
: <color> The text outline colour. Requires font_settings$sdf = TRUE.

font_settings
: <font_settings> Advanced options for fine tuning the appearance and performance of the generated font_atlas.

word_break
: <"break-word" | "break-all"> Requires a valid max_width.

max_width
: <number> Used together with word_break for wrapping text. Specifies the width limit to break the text into multiple lines.

get_text
: <accessor> The text value of each text label. Accepts a tidy-eval character column of labels.

get_position
: <accessor> The feature positions. A <point/multipoint> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_color
: <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_size
: <accessor | scale | number> The font size of each text label, in units specified by size_units. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

get_angle
: <accessor | number> The rotating angle of each text label in degrees. Accepts a single numeric value, or a tidy-eval column of numbers.

get_text_anchor
: <accessor | "start" | "middle" | "end"> The text label anchor. May be a single value, or a tidy-eval character column.

get_alignment_baseline
: <accessor | "top" | "center" | "bottom"> The text label alignment baseline. May be a single value, or a tidy-eval character column.

get_pixel_offset
: <accessor | number> The pixel offset for each object. Accepts a single length-2 numeric vector, or a tidy-eval list column.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle <boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/layers/text-layer.md

Other core-layers: arc_layer, bitmap_layer, column_layer, geojson_layer, grid_cell_layer, icon_layer, line_layer, path_layer, point_cloud_layer, polygon_layer, scatterplot_layer, solid_polygon_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, tile_3d_layer, tile_layer, trips_layer

---

tile_3d_layer *Tile 3D Layer*

Description

Tile 3D Layer

Usage

```
add_tile_3d_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "Tile3DLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
```

```
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_point_color = "#000000ff",
  point_size = 1,
  loader = list(id = "3d-tiles", name = "3D Tiles", module = "3d-tiles", version =
    "3.2.13", extensions = c("cmpt", "pnts", "b3dm", "i3dm"), mimeTypes =
   c("application/octet-stream"), tests = c("cmpt", "pnts", "b3dm", "i3dm"), parse =
   NULL, options = list(`3d-tiles` = list(loadGLTF = TRUE, decodeQuantizedPositions =
    FALSE, isTileset = "auto", assetGltfUpAxis = NULL))),
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL
)

update_tile_3d_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_point_color = cur_value(),
  point_size = cur_value(),
  loader = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same

type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.

name <string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

group_name <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.

data <data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.

visible <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

pickable <boolean> Determines if the layer responds to pointer / touch events.

opacity <number> Determines the layer's opacity.

wrap_longitude <boolean> Normalises geometry longitudes.

position_format
<"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
<accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

get_point_color
<color> The colour of each object.

point_size <number> The radius of all points in units specified by size_units.

loader object

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/tile3d-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, s2_layer, terrain_layer, tile_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_layer, trips_layer

---

tile_json *Tile json*

Description

Intended for use as the data parameter of mvt_layer(). Creates a tile json url from a tileset_id and tile_service. Tile service defaults to getOption("rdeck.tile_service") %||% "mapbox".

The created url will be fetched and parsed with jsonlite::fromJSON().

Usage

```
tile_json(tileset_id, tile_service = NULL)
```

Arguments

tileset_id	<string> The tileset_id, may be one of the following • a tileset identifier, e.g. "mapbox.mapbox-streets-v8" • a tileset identifier.json, e.g. "mapbox.mapbox-streets-v8.json" • a url, e.g. "mapbox://mapbox.mapbox-streets-v8", or "https://mytileserver/tileset" tile_service is unused if a url is supplied.
tile_service	<string> The tile service name, defaults to 'getOption("rdeck.tile_service") %||% "mapbox".

Note

Authentication via mapbox_access_token() occurs when tile_service = "mapbox", or tileset_id uses the mapbox scheme (i.e. mapbox://).

Examples

```
## Not run:
tile_json("mapbox.mapbox-streets-v8", "mapbox")
tile_json("mapbox.mapbox-streets-v8.json", "mapbox")
tile_json("mapbox://mapbox.mapbox-streets-v8")
tile_json("mapbox://mapbox.mapbox-streets-v8.json")
tile_json("https://mytileserver/tileset.json")
tile_json("tileset.json", "https://mytileserver")

## End(Not run)
```

---

tile_layer *Tile Layer*

Description

Tile Layer

Usage

```
add_tile_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "TileLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  get_tile_data = NULL,
  extent = NULL,
  tile_size = 512,
  max_zoom = NULL,
  min_zoom = 0,
  max_cache_size = NULL,
  max_cache_byte_size = NULL,
  refinement_strategy = "best-available",
  z_range = NULL,
  max_requests = 6,
  zoom_offset = 0,
```

```
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL,
  desaturate = 0,
  transparent_color = "#00000000",
  tint_color = "#ffffff"
)

update_tile_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  get_tile_data = cur_value(),
  extent = cur_value(),
  tile_size = cur_value(),
  max_zoom = cur_value(),
  min_zoom = cur_value(),
  max_cache_size = cur_value(),
  max_cache_byte_size = cur_value(),
  refinement_strategy = cur_value(),
  z_range = cur_value(),
  max_requests = cur_value(),
  zoom_offset = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value(),
  desaturate = cur_value(),
  transparent_color = cur_value(),
  tint_color = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.

`id` <string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to `uuid::UUIDgenerate()`, but should be explicitly defined for updatable layers in a shiny application.

`name` <string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.

`group_name` <string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if `visibility_toggle = TRUE`.

`data` <character> A character vector of raster tile url templates. Substrings `"{x}"`, `"{y}"`, `"{z}"` will be replaced with a tile's actual index on request.

If multiple url templates are supplied, each endpoint must return the same data for the same tile index.

`visible` <boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.

`pickable` <boolean> Determines if the layer responds to pointer / touch events.

`opacity` <number> Determines the layer's opacity.

`wrap_longitude` <boolean> Normalises geometry longitudes.

`position_format`

<`"XY"` | `"XYZ"`> Determines whether each coordinate has two (XY) or three (XYZ) elements.

`color_format` <`"RGB"` | `"RGBA"`> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

`auto_highlight` <boolean> When `TRUE`, the current object *hovered* by the cursor is highlighted by `highlight_color`.

`highlight_color`

<accessor | scale | color> When `auto_highlight` and `pickable` are enabled, `highlight_color` determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

`get_tile_data` <JS> retrieves the data of each tile. See deck.gl getTileData.

`extent` <`c(min_x, min_y, max_x, max_y)`> Tiles in this bounding box will be rendered at `min_zoom`, when zoomed out below `min_zoom`.

`tile_size` <number> A power of 2 that is the pixel dimensions of the tile.

`max_zoom` <number> Tiles above this zoom level are not shown. Defaults to `NULL`.

`min_zoom` <number> Tiles below this zoom level are not shown. Defaults to `0`.

`max_cache_size` <number> Maximum number of tiles that can be cached. Defaults to 5x the number of tiles in current viewport.

`max_cache_byte_size`

<number> Maximum memory used for caching tiles.

`refinement_strategy`

<`"best-available"` | `"no-overlap"` | `"never"`> How the tile layer refines visibility of tiles. Defaults to `"best-available"`.

`z_range` <`c(min, max)`> Array representing the range of heights in the tile.

max_requests <number> Maximum number of concurrent HTTP requests across all specified tile provider domains. If a negative number is supplied no throttling occurs (HTTP/2 only).

zoom_offset <int> The offset changes the zoom level at which the tiles are fetched.

blending_mode <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

- normal: Normal blending doesn't alter colours of overlapping objects.
- additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.
- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
<boolean> Whether this layer will appear in the layer selector.

tooltip <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

desaturate <number> The desaturation of the bitmap. Between 0 and 1, being the original colour, 1 being greyscale.

transparent_color
<color> The colour to use for transparent pixels.

tint_color <color> The colour to tint the bitmap by. Alpha channel is ignored if supplied.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/tile-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, s2_layer, terrain_layer, tile_3d_layer, trips_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, trips_layer

---

trips_layer *Trips Layer*

Description

Trips Layer

Usage

```
add_trips_layer(
  rdeck,
  ...,
  id = uuid::UUIDgenerate(),
  name = "TripsLayer",
  group_name = NULL,
  data = NULL,
  visible = TRUE,
  pickable = FALSE,
  opacity = 1,
  wrap_longitude = FALSE,
  position_format = "XY",
  color_format = "RGBA",
  auto_highlight = FALSE,
  highlight_color = "#00008080",
  width_units = "meters",
  width_scale = 1,
  width_min_pixels = 0,
  width_max_pixels = 9007199254740991,
  joint_rounded = FALSE,
  cap_rounded = FALSE,
  miter_limit = 4,
  billboard = FALSE,
  get_path = path,
  get_color = "#000000ff",
  get_width = 1,
  fade_trail = TRUE,
  trail_length = 120,
  get_timestamps = timestamps,
  blending_mode = "normal",
  visibility_toggle = TRUE,
  tooltip = NULL,
  loop_length = 1800,
  animation_speed = 30
)

update_trips_layer(
  rdeck,
  ...,
  id,
  name = cur_value(),
  group_name = cur_value(),
  data = cur_value(),
  visible = cur_value(),
  pickable = cur_value(),
  opacity = cur_value(),
  wrap_longitude = cur_value(),
```

```
  position_format = cur_value(),
  color_format = cur_value(),
  auto_highlight = cur_value(),
  highlight_color = cur_value(),
  width_units = cur_value(),
  width_scale = cur_value(),
  width_min_pixels = cur_value(),
  width_max_pixels = cur_value(),
  joint_rounded = cur_value(),
  cap_rounded = cur_value(),
  miter_limit = cur_value(),
  billboard = cur_value(),
  get_path = cur_value(),
  get_color = cur_value(),
  get_width = cur_value(),
  fade_trail = cur_value(),
  trail_length = cur_value(),
  get_timestamps = cur_value(),
  blending_mode = cur_value(),
  visibility_toggle = cur_value(),
  tooltip = cur_value(),
  loop_length = cur_value(),
  animation_speed = cur_value()
)
```

Arguments

rdeck	<rdeck | rdeck_proxy> An rdeck map instance.
...	Additional parameters that will be forwarded to deck.gl javascript without validation nor processing. All dots must be named and will be camelCased when serialised. A warning is raised when dots are used, warning class rdeck_dots_nonempty.
id	<string> The layer's identifier must be unique for among all layers of the same type for a map. Defaults to uuid::UUIDgenerate(), but should be explicitly defined for updatable layers in a shiny application.
name	<string> Identifies the layer on tooltips and legends. It does not need to be unique, but should be brief. Defaults to the deck.gl class name for the layer.
group_name	<string> Defines the group that this layer belongs to. Currently only effective on the layer selector, if visibility_toggle = TRUE.
data	<data.frame | sf | string> The layer's data. Data frames and sf objects will contain all columns that are referenced by the layer's accessors. Strings will be interpreted as a URL and data will be retrieved dynamically in the browser.
visible	<boolean> Determines whether the layer is visible or not; also determines whether any legend elements for the layer will be displayed.
pickable	<boolean> Determines if the layer responds to pointer / touch events.
opacity	<number> Determines the layer's opacity.
wrap_longitude	<boolean> Normalises geometry longitudes.

position_format
: <"XY" | "XYZ"> Determines whether each coordinate has two (XY) or three (XYZ) elements.

color_format
: <"RGB" | "RGBA"> Determines whether the alpha channel of the colours will be ignored by accessors, making all colours opaque.

auto_highlight
: <boolean> When TRUE, the current object *hovered* by the cursor is highlighted by highlight_color.

highlight_color
: <accessor | scale | color> When auto_highlight and pickable are enabled, highlight_color determines the colour of the currently highlighted object. If a single colour value is supplied, that colour will be used to highlight all objects in the layer. Per-object highlighting is achieved with a colour scale, or a tidy-eval column of colours.

width_units
: <"pixels" | "common" | "meters"> The units of the line_width.

width_scale
: <number> The scaling multiplier for the width of each line.

width_min_pixels
: <number> The minimum line width in pixels.

width_max_pixels
: <number> The maximum line width in pixels.

joint_rounded
: <boolean> If TRUE, draw round joints; else draw square joints.

cap_rounded
: <boolean> If TRUE, draw round caps; else draw square caps.

miter_limit
: <number> The maximum extent of a joint in ratio to the stroke width. Only applicable if rounded == FALSE.

billboard
: <boolean> If TRUE, extrude the path in screen space (width always faces) the camera; if FALSE, the width always faces up (z).

get_path
: <accessor> The feature paths. A <linestring/multilinestring> wk-geometry column with CRS EPSG:4326. Supports tidy-eval.

get_color
: <accessor | scale | color> The colour of each object. Accepts a single colour value, a colour scale, or a tidy-eval column of colours.

get_width
: <accessor | scale | number> The width of each object, in units specified by width_scale. Accepts a single numeric value, a numeric scale, or a tidy-eval column of numbers.

fade_trail
: <boolean> Whether or not the path fades out.

trail_length
: <number> The number of seconds for a path to completely fade out.

get_timestamps
: <accessor> The timestamps for each *trip*. For each trip, each timestamp corresponds to a position in the linestring returned by get_path, representing the time at which that position was visited.

  Accepts a tidy-eval list column of numbers.

blending_mode
: <"normal" | "additive" | "subtractive"> Sets the blending mode. Blending modes:

  - normal: Normal blending doesn't alter colours of overlapping objects.
  - additive: Additive blending adds colours of overlapping objects. Useful for highlighting dot density on dark maps.

- subtractive: Subtractive blending darkens overlapping objects. Useful for highlighting dot density on light maps.

visibility_toggle
: <boolean> Whether this layer will appear in the layer selector.

tooltip
: <tooltip> Defines the columns (and their order) that will be displayed in the layer tooltip, if pickable == TRUE. Supports tidy-select if a data is a data.frame. Geometry columns are always removed.

loop_length
: <number> The number of seconds to complete an animation loop.

animation_speed
: <number> The animation speed multiplier.

See Also

https://github.com/visgl/deck.gl/blob/8.7-release/docs/api-reference/geo-layers/trips-layer.md

Other geo-layers: great_circle_layer, h3_cluster_layer, h3_hexagon_layer, mvt_layer, quadkey_layer, s2_layer, terrain_layer, tile_3d_layer, tile_layer

Other layers: arc_layer, bitmap_layer, column_layer, contour_layer, cpu_grid_layer, geojson_layer, gpu_grid_layer, great_circle_layer, grid_cell_layer, grid_layer, h3_cluster_layer, h3_hexagon_layer, heatmap_layer, hexagon_layer, icon_layer, line_layer, mvt_layer, path_layer, point_cloud_layer, polygon_layer, quadkey_layer, s2_layer, scatterplot_layer, scenegraph_layer, screen_grid_layer, simple_mesh_layer, solid_polygon_layer, terrain_layer, text_layer, tile_3d_layer, tile_layer

---

view_state *Create a view_state*

Description

Create a view_state

Usage

```
view_state(center = c(0, 0), zoom = 0, pitch = 0, bearing = 0, ...)
```

Arguments

center
: numeric | sf::st_point Center of the viewport on map

zoom
: numeric Scale = 2^{zoom}

pitch
: numeric Camera angle in degrees (0 is straight down)

bearing
: numeric Map rotation in degrees (0 means north is up)

...
: additional parameters to pass to view_state | initial_view_state

See Also

https://github.com/uber-archive/viewport-mercator-project/blob/master/docs/api-reference/web-mercator-viewport.md

xy *Point vectors*

Description

Create point vectors

Usage

```
xy(x = double(), y = double(), crs = wk::wk_crs_longlat())

xyz(x = double(), y = double(), z = double(), crs = wk::wk_crs_longlat())

sfc_point(x = double(), y = double(), z = NULL, crs = wk::wk_crs_longlat())
```

Arguments

x	<double> Coordinate x dimension
y	<double> Coordinate y dimension
crs	A value to be propagated as the CRS for this vector.
z	<double> Coordinate z dimension

Value

A vector of coordinate values.

Functions

- `xy()`: Efficient 2-dimensional point vector
- `xyz()`: Efficient 3-dimensional point vector
- `sfc_point()`: Simple features point vector

Examples

```
xy(1:5, 1:5)
xyz(1:5, 1:5, 1:5)
sfc_point(1:5, 1:5)
sfc_point(1:5, 1:5, 1:5)
```

Index