Package: maplamina (via r-universe)

May 22, 2026

Title High-Performance 'WebGL' Mapping Widgets for R

Version 0.1.0

Description Creates interactive maps using 'MapLibre GL' and 'deck.gl' via 'htmlwidgets'. Provides GPU-accelerated layers for points, lines and polygons, plus linked user interface components such as filters, views and summary cards for exploratory analysis and production dashboards.

License MIT + file LICENSE

URL https://github.com/jhumbl/maplamina, https://jhumbl.github.io/maplamina/

BugReports https://github.com/jhumbl/maplamina/issues

Depends R (>= 4.1.0)

Imports htmlwidgets, digest, base64enc, sf

Suggests testthat (>= 3.0.0), jsonlite, knitr, rmarkdown

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-03-15 16:36:56 UTC

RemoteUrl https://github.com/jhumbl/maplamina

RemoteRef HEAD

RemoteSha d695f4055e0fd257325eb49e8220296039e10e5c

Contents

`add_circles` *Add a circle layer*

Description

Adds a GPU-rendered circle layer (points). Use formulas (e.g. `~mag * 2`) to map columns to aesthetics.

Usage

```
add_circles(
  map,
  data = NULL,
  lon = NULL,
  lat = NULL,
  color = "darkblue",
  opacity = 1,
  width = 1,
  fill_color = "dodgerblue",
  fill_opacity = 0.8,
  radius = 6,
  stroke = TRUE,
  tooltip = NULL,
  popup = NULL,
  id = NULL,
  group = NULL,
  pickable = TRUE,
  radius_units = c("pixels", "meters", "common"),
  radius_min_pixels = NULL,
  radius_max_pixels = NULL,
  width_units = c("pixels", "meters", "common"),
  width_min_pixels = NULL,
  width_max_pixels = NULL
)
```

Arguments

map	A maplamina widget created by maplamina().
data	Data for this layer. If NULL, uses the default data supplied to maplamina().
lon, lat	Longitude/latitude aesthetics (formula or scalar). Required for non-sf data.
color, opacity, width	Stroke color/opacity/width.
fill_color, fill_opacity	Fill color/opacity. fill_color can be a single color, a vector of colors, or a color scale spec such as color_quantile().
radius	Circle radius (numeric or formula).
stroke	Logical; draw circle stroke.
tooltip, popup	Optional tmpl() objects (or NULL) for hover/click content.
id, group	Optional layer id and group name.
pickable	Logical; whether features can be picked (tooltip/popup).
radius_units	Units for radius; one of "pixels", "meters", or "common".
radius_min_pixels, radius_max_pixels	Optional clamp in pixels when using meter/common units.
width_units	Units for stroke width; one of "pixels", "meters", or "common".

`width_min_pixels`, `width_max_pixels`
: Optional clamp in pixels when using meter/common units.

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d, radius = ~value, fill_color = "dodgerblue")
```

---

add_filters *Add filters to a layer*

Description

Registers one or more filters for a layer. When multiple layers share the same `bind`, controls merge across layers by (bind, label, type).

Usage

```
add_filters(map, ..., id = NULL, bind = NULL, position = NULL, layer_id = NULL)
```

Arguments

`map`	A maplamina widget created by `maplamina()`.
`...`	One or more `filter_range()` / `filter_select()` objects (or a single list of them).
`id`	Optional id used as a shorthand bind id when `bind` is omitted.
`bind`	Bind group id for shared UI control.
`position`	Optional UI position hint (applied to the control group).
`layer_id`	Target layer id (defaults to the most recently added layer).

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d) |>
  add_filters(
    filter_range(~value, default = c(4, 6)),
    bind = "filters"
  )
```

---

add_fullscreen *Add MapLibre fullscreen control*

Description

Add MapLibre fullscreen control

Usage

```
add_fullscreen(map, position = "topright", ...)
```

Arguments

map	A maplamina widget created by maplamina().
position	Control position: "topleft", "topright", "bottomleft", "bottomright".
...	Named options passed through to the underlying MapLibre control.

Value

The modified map widget.

Examples

```
maplamina() |>
  add_fullscreen()
```

add_geolocate *Add MapLibre geolocate control*

Description

Add MapLibre geolocate control

Usage

```
add_geolocate(map, position = "topright", track_user_location = FALSE, ...)
```

Arguments

map	A maplamina widget created by maplamina().
position	Control position: "topleft", "topright", "bottomleft", "bottomright".
track_user_location	Logical; keep tracking after initial locate.
...	Named options passed through to the underlying MapLibre control.

Value

The modified map widget.

Examples

```
maplamina() |>
  add_geolocate()
```

add_icons *Add an icon layer*

Description

Adds a point icon layer. Icons are referenced by id (resolved in the frontend icon set).

Usage

```
add_icons(
  map,
  data = NULL,
  lon = NULL,
  lat = NULL,
  icon = "marker",
  size = 18,
  color = "#3388ff",
```

```
  opacity = 1,
  tooltip = NULL,
  popup = NULL,
  id = NULL,
  group = NULL,
  pickable = TRUE,
  size_units = c("pixels", "meters", "common"),
  size_min_pixels = NULL,
  size_max_pixels = 64,
  icon_anchor = NULL,
  mask = TRUE,
  occlude = FALSE
)
```

Arguments

map	A maplamina widget created by maplamina().
data	Data for this layer. If NULL, uses the default data supplied to maplamina().
lon, lat	Longitude/latitude aesthetics (formula or scalar). Required for non-sf data.
icon	Icon id string (e.g. "marker").
size, color, opacity	Icon size and color/opacity (can be formulas).
tooltip, popup	Optional tmpl() objects (or NULL) for hover/click content.
id, group	Optional layer id and group name.
pickable	Logical; whether features can be picked (tooltip/popup).
size_units	Units for size; one of "pixels", "meters", or "common".
size_min_pixels, size_max_pixels	Optional clamp in pixels when using meter/common units.
icon_anchor	Optional anchor (frontend-specific).
mask	Logical; whether to mask the icon.
occlude	Logical; whether icons occlude each other.

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_icons(d, icon = "star", size = 20)
```

add_legend *Add a legend component*

Description

Adds a categorical or continuous legend. Legends can be conditionally displayed based on layer/view, and can be mounted via the panel by `bind`.

Usage

```
add_legend(
  map,
  title = NULL,
  type = c("categorical", "continuous"),
  position = c("bottomleft", "bottomright", "topleft", "topright"),
  values = NULL,
  colors = NULL,
  shapes = "square",
  sizes = NULL,
  range = NULL,
  labels = NULL,
  breaks = NULL,
  gradient = NULL,
  shape = "square",
  size = 12,
  layer = NULL,
  view = NULL,
  bind = NULL,
  id = NULL
)
```

Arguments

`map`	A maplamina widget created by `maplamina()`.
`title`	Optional legend title.
`type`	Legend type: `"categorical"` or `"continuous"`.
`position`	Legend position hint: `"bottomleft"`, `"bottomright"`, `"topleft"`, `"topright"`.
`values, colors`	Categorical legend labels and colors (same length).
`shapes, sizes`	Optional categorical shapes/sizes (length 1 or length(values)).
`range, labels, breaks`	Continuous legend range/labels/breaks.
`gradient`	Vector of 2+ colors for the continuous legend.
`shape, size`	Continuous legend swatch/shape and size.
`layer, view`	Optional conditional display rule.
`bind`	Bind group id for mounting in the panel (defaults to legend id).
`id`	Optional component id (otherwise generated).

Value

The modified map widget.

Examples

```
maplamina() |>
  add_legend(
    title = "Magnitude",
    type = "continuous",
    range = c(3, 7),
    gradient = c("lightyellow", "red")
  )
```

---

add_lines *Add a line layer*

Description

Adds a GPU-rendered line/path layer.

Usage

```
add_lines(
  map,
  data = NULL,
  color = "#3388ff",
  opacity = 1,
  width = 1,
  tooltip = NULL,
  popup = NULL,
  id = NULL,
  group = NULL,
  pickable = TRUE,
  width_units = c("pixels", "meters", "common"),
  width_min_pixels = NULL,
  width_max_pixels = NULL
)
```

Arguments

map	A maplamina widget created by maplamina().
data	Data for this layer. Typically an sf object with LINESTRING geometry.
color, opacity, width	Line color/opacity/width.
tooltip, popup	Optional tmpl() objects (or NULL) for hover/click content.
id, group	Optional layer id and group name.

pickable	Logical; whether features can be picked (tooltip/popup).
width_units	Units for width; one of "meters", "pixels", or "common".
width_min_pixels, width_max_pixels	Optional clamp in pixels when using meter/common units.

Value

The modified map widget.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  ln <- sf::st_sfc(
    sf::st_linestring(matrix(c(-122.4, 37.8, -122.5, 37.85), ncol = 2, byrow = TRUE)),
    crs = 4326
  )
  ln <- sf::st_sf(id = 1, geometry = ln)

  maplamina(ln) |>
    add_lines(color = "black", width = 3)
}
```

---

add_markers *Add a marker layer*

Description

Convenience wrapper around add_icons() using the default marker icons.

Usage

```
add_markers(
  map,
  data = NULL,
  lon = NULL,
  lat = NULL,
  size = 18,
  color = "dodgerblue",
  opacity = 1,
  tooltip = NULL,
  popup = NULL,
  id = NULL,
  group = NULL,
  pickable = TRUE,
  size_units = c("pixels", "meters", "common"),
  size_min_pixels = NULL,
  size_max_pixels = NULL
)
```

Arguments

map	A maplamina widget created by maplamina().
data	Data for this layer. If NULL, uses the default data supplied to maplamina().
lon, lat	Longitude/latitude aesthetics (formula or scalar). Required for non-sf data.
size, color, opacity	Marker size and color/opacity (can be formulas).
tooltip, popup	Optional tmpl() objects (or NULL) for hover/click content.
id, group	Optional layer id and group name.
pickable	Logical; whether features can be picked (tooltip/popup).
size_units	Units for size; one of "pixels", "meters", or "common".
size_min_pixels, size_max_pixels	Optional clamp in pixels when using meter/common units.

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_markers(d, size = ~value * 3)
```

---

add_navigation *Add MapLibre navigation controls*

Description

Adds zoom buttons and/or compass control.

Usage

```
add_navigation(
  map,
  position = "topright",
  compass = TRUE,
  zoom_controls = TRUE,
  ...
)
```

Arguments

map	A maplamina widget created by maplamina().
position	Control position: "topleft", "topright", "bottomleft", "bottomright".
compass	Logical; show compass control.
zoom_controls	Logical; show zoom buttons.
...	Named options passed through to the underlying MapLibre control.

Value

The modified map widget.

Examples

```
maplamina() |>
  add_navigation(position = "topright", compass = FALSE)
```

---

add_panel *Add a global, presentation-only panel*

Description

The panel mounts previously-declared controls by bind id (see section() and sections()).

Usage

```
add_panel(
  map,
  title = NULL,
  description = NULL,
  icon = NULL,
  dividers = TRUE,
  position = NULL,
  sections = NULL
)
```

Arguments

map	A maplamina widget created by maplamina().
title	Panel title.
description	Optional panel description.
icon	Optional URL for a small icon shown beside the title.
dividers	Logical; whether to render divider lines between sections.
position	Optional corner position for the panel container: "topleft", "topright", "bottomleft", "bottomright".
sections	Panel layout, created with sections(section("id"), ...).

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d) |>
  add_filters(filter_range(~value), bind = "filters") |>
  add_panel(sections = sections(section("filters")))
```

---

add_polygons *Add a polygon layer*

Description

Adds a polygon layer (fill + optional stroke). For data-driven fills, use a color scale spec such as `color_quantile()` or `color_factor()`.

Usage

```
add_polygons(
  map,
  data = NULL,
  color = "darkblue",
  opacity = 1,
  width = 1,
  fill_color = "dodgerblue",
  fill_opacity = 0.6,
  stroke = TRUE,
  elevation = NULL,
  elevation_scale = 1,
  tooltip = NULL,
  popup = NULL,
  id = NULL,
  group = NULL,
  pickable = TRUE,
  width_units = c("pixels", "meters", "common"),
  width_min_pixels = NULL,
  width_max_pixels = NULL
)
```

Arguments

map	A maplamina widget created by maplamina().
data	Data for this layer. Typically an sf object with POLYGON/MULTIPOLYGON geometry.
color, opacity, width	Stroke color/opacity/width.
fill_color, fill_opacity	Fill color/opacity. fill_color can be a single color, a vector of colors, or a color scale spec such as color_quantile().
stroke	Logical; draw polygon stroke.
elevation, elevation_scale	Optional extrusion height (numeric or formula) and scale.
tooltip, popup	Optional tmpl() objects (or NULL) for hover/click content.
id, group	Optional layer id and group name.
pickable	Logical; whether features can be picked (tooltip/popup).
width_units	Units for stroke width; one of "pixels", "meters", or "common".
width_min_pixels, width_max_pixels	Optional clamp in pixels when using meter/common units.

Value

The modified map widget.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
  maplamina(nc) |>
    add_polygons(fill_color = color_quantile(~BIR74), stroke = FALSE)
}
```

---

add_scalebar *Add MapLibre scale bar*

Description

Add MapLibre scale bar

Usage

```
add_scalebar(
  map,
  position = "bottomleft",
  unit = c("metric", "imperial", "nautical"),
  max_width = 100,
  ...
)
```

Arguments

map	A maplamina widget created by maplamina().
position	Control position: "topleft", "topright", "bottomleft", "bottomright".
unit	Units for the scale bar.
max_width	Maximum scale bar width (pixels).
...	Named options passed through to the underlying MapLibre control.

Value

The modified map widget.

Examples

```
maplamina() |>
  add_scalebar(unit = "metric")
```

---

add_summaries *Add summaries to a layer*

Description

Registers one or more summary rows that update with filtering. When multiple layers share the same bind, a single summaries card can be created.

Usage

```
add_summaries(
  map,
  ...,
  id = NULL,
  bind = NULL,
  position = NULL,
  layer_id = NULL
)
```

Arguments

map	A maplamina widget created by maplamina().
...	One or more summary_*() objects (or a single list of them).
id	Optional id used as a shorthand bind id when bind is omitted.
bind	Bind group id for shared UI control.
position	Optional UI position hint (applied to the control group).
layer_id	Target layer id (defaults to the most recently added layer).

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d) |>
  add_filters(filter_range(~value), bind = "filters") |>
  add_summaries(summary_mean(~value, label = "Avg value"), bind = "summaries") |>
  add_panel(sections = sections(section("filters"), section("summaries")))
```

---

add_views *Add views to a layer*

Description

Registers a per-layer views component. When multiple layers share the same `bind`, a single selector control is created.

Usage

```
add_views(
  map,
  ...,
  id = NULL,
  bind = NULL,
  position = NULL,
  layer_id = NULL,
  duration = 750,
 easing = c("smoothstep", "linear", "easein", "easeout", "easeinout", "easeInOutCubic")
)
```

Arguments

`map`	A maplamina widget created by `maplamina()`.
`...`	One or more `view()` objects (or a single list of them).
`id`	Optional component id (also used as the default bind id).
`bind`	Bind group id for shared UI control. If omitted, defaults to `id` (or an auto id).
`position`	Optional UI position hint (applied to the control group).
`layer_id`	Target layer id (defaults to the most recently added layer).
`duration`	Animation duration (ms) for switching views.
`easing`	Easing function name for switching views.

Value

The modified map widget.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d, radius = 5) |>
  add_views(
    view("magnitude", radius = ~value * 3),
    view("faint", fill_opacity = 0.2)
  )
```

---

base_tiles *Basemap tile styles*

Description

A named list of MapLibre-compatible style URLs for common open-source basemaps. Pass any entry directly to the `style` argument of `maplamina`.

Usage

```
base_tiles
```

Format

A named list of character strings (URLs).

carto_positron Light basemap with labels. Good default for thematic maps.

carto_positron_nolabels Light basemap without labels. Clean background for dense data.

carto_dark_matter Dark basemap with labels. Good for glowing point layers.

carto_dark_matter_nolabels Dark basemap without labels.

carto_voyager Colored basemap with labels. More geographic detail than Positron.

carto_voyager_nolabels Colored basemap without labels.

openfreemap_liberty OSM Liberty style hosted by OpenFreeMap. No API key required.

openfreemap_bright OSM Bright style hosted by OpenFreeMap. No API key required.

openfreemap_positron Positron style hosted by OpenFreeMap. No API key required.

demotiles MapLibre demo tiles. Lightweight, no API key, useful for testing.

Examples

```
# Polygons example (sf ships the nc shapefile)
if (requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)

  maplamina(nc) |>
    add_polygons(fill_color = color_quantile(~BIR74), stroke = FALSE)

  maplamina(nc, style = base_tiles$carto_dark_matter) |>
    add_polygons(fill_color = color_quantile(~BIR74, "Inferno"), stroke = FALSE)
}
```

`color_bin` *Binned (equal-interval or manual breaks) color scale specification*

Description

Binned (equal-interval or manual breaks) color scale specification

Usage

```
color_bin(
  expr,
  palette = NULL,
  bins = 5,
  domain = NULL,
  na_color = "#00000000",
  reverse = FALSE,
  clamp = TRUE
)
```

Arguments

`expr`	A formula selecting a numeric column (e.g. `~value`).
`palette`	`NULL`, a palette name, or a vector of colors.
`bins`	Number of bins, or a numeric vector of breakpoints.
`domain`	Optional numeric range `c(min, max)`; defaults to data range.
`na_color`	Color used for missing values.
`reverse`	Logical; reverse the palette.
`clamp`	Logical; if `TRUE`, clamp values outside `domain` to the ends.

Value

A color scale specification object.

`color_factor` *Categorical color scale specification*

Description

Categorical color scale specification

Usage

```
color_factor(
  expr,
  palette = NULL,
  domain = NULL,
  na_color = "#00000000",
  reverse = FALSE
)
```

Arguments

`expr`	A formula selecting a categorical column (e.g. `~region`).
`palette`	`NULL`, a palette name, or a vector of colors.
`domain`	Optional character vector of levels (controls ordering and which levels are shown).
`na_color`	Color used for missing values.
`reverse`	Logical; reverse the palette.

Value

A color scale specification object.

`color_numeric` *Continuous color scale specification*

Description

Creates a numeric color scale spec that is resolved during widget compilation. Use in `color`/`fill_color` aesthetics (e.g. `fill_color = color_numeric(~x)`).

Usage

```
color_numeric(
  expr,
  palette = NULL,
  domain = NULL,
  steps = 256L,
  na_color = "#00000000",
  reverse = FALSE,
  clamp = TRUE
)
```

Arguments

expr	A formula selecting a numeric column (e.g. ~value).
palette	NULL, a palette name (passed to grDevices::hcl.colors()), or a vector of colors.
domain	Optional numeric range c(min, max); defaults to data range.
steps	Number of palette steps for interpolation.
na_color	Color used for missing values.
reverse	Logical; reverse the palette.
clamp	Logical; if TRUE, clamp values outside domain to the ends.

Value

A color scale specification object.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
  maplamina(nc) |>
    add_polygons(fill_color = color_numeric(~BIR74))
}
```

---

color_quantile *Quantile color scale specification*

Description

Quantile color scale specification

Usage

```
color_quantile(
  expr,
  palette = NULL,
  n = 5,
  domain = NULL,
  na_color = "#00000000",
  reverse = FALSE,
  clamp = TRUE
)
```

Arguments

`expr`	A formula selecting a numeric column (e.g. `~value`).
`palette`	`NULL`, a palette name, or a vector of colors.
`n`	Number of quantile bins.
`domain`	Optional numeric range `c(min, max)`; defaults to data range.
`na_color`	Color used for missing values.
`reverse`	Logical; reverse the palette.
`clamp`	Logical; if `TRUE`, clamp values outside `domain` to the ends.

Value

A color scale specification object.

`filter_range` *Define a range filter*

Description

Creates a numeric range filter specification to be registered with `add_filters()`.

Usage

```
filter_range(
  col,
  label = NULL,
  default = NULL,
  min = NULL,
  max = NULL,
  step = NULL,
  live = TRUE,
  id = NULL
)
```

Arguments

col	A formula selecting a numeric column (e.g. ~mag).
label	Optional label for the UI control (defaults to the column name).
default	Optional default range c(min, max) (length 2).
min, max	Optional explicit min/max (otherwise computed from data).
step	Optional step size for the UI slider.
live	Logical; update the map continuously while dragging.
id	Optional filter id (otherwise generated).

Value

A filter specification object.

Examples

```
f <- filter_range(~mpg, default = c(15, 30))
f
```

---

filter_select *Define a select filter*

Description

Creates a categorical filter specification to be registered with add_filters().

Usage

```
filter_select(
  col,
  label = NULL,
  multi = TRUE,
  dropdown = NULL,
  searchable = TRUE,
  default = NULL,
  max_levels = NULL,
  id = NULL
)
```

Arguments

col	A formula selecting a column (e.g. ~region).
label	Optional label for the UI control (defaults to the column name).
multi	Logical; allow multiple selections.
dropdown	Optional UI hint (frontend-specific).

searchable	Logical; allow searching within options.
default	Optional default selection(s).
max_levels	Optional maximum number of levels to show (frontend may truncate).
id	Optional filter id (otherwise generated).

Value

A filter specification object.

Examples

```
f <- filter_select(~Species, default = c("setosa", "versicolor"))
f
```

`maplamina` *Create a Maplamina map widget*

Description

Creates an interactive MapLibre + deck.gl map widget. You can add layers and UI components (views/filters/summaries/panel) with the `add_*()` functions.

Usage

```
maplamina(
  data = NULL,
  style = "https://basemaps.cartocdn.com/gl/positron-gl-style/style.json",
  projection = c("mercator", "globe"),
  dragRotate = FALSE,
  fit_bounds = TRUE,
  show_layer_controls = TRUE,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

data	Optional default dataset used by subsequent `add_*()` calls when their `data` argument is `NULL`.
style	MapLibre style URL (or a named style from `base_tiles`).
projection	Map projection; one of `"mercator"` or `"globe"`.
dragRotate	Logical; whether drag-rotate is enabled (also affects compass).
fit_bounds	Logical; whether the map initially fits the bounds of all layers.
show_layer_controls	Logical; show built-in per-layer visibility controls.

width, height	Widget width/height (CSS units or numeric pixels). If height is NULL, a viewer-friendly default is used.
elementId	Optional HTML element id.

Value

An htmlwidget maplamina widget.

Examples

```
# Minimal widget
maplamina()

# With a Circle layer
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d)
```

---

section *Create a panel section that mounts a control by bind id*

Description

Panel sections refer to bind ids created by components such as add_views(), add_filters(), and add_summaries().

Usage

```
section(id)
```

Arguments

id	Bind id (e.g. the bind supplied to add_views/add_filters, or the component id when bind is omitted).

Value

A panel section object.

Examples

```
section("filters")
```

sections *Create a list of panel sections*

Description

Create a list of panel sections

Usage

```
sections(...)
```

Arguments

`...`	One or more `section()` objects, character ids, or lists of sections.

Value

A panel sections container.

Examples

```
sections(section("views"), section("filters"))
```

summary_count *Summary: count rows*

Description

Counts the number of filtered rows.

Usage

```
summary_count(
  col = NULL,
  label = NULL,
  digits = NULL,
  prefix = NULL,
  suffix = NULL,
  id = NULL
)
```

Arguments

col	Optional formula like ~id (validated if provided).
label	Display label for the summary row.
digits	Optional number of digits for formatting (applied in JS).
prefix, suffix	Optional prefix/suffix strings (applied in JS).
id	Optional summary id.

Value

A summary specification object.

---

summary_max *Summary: max value*

Description

Summary: max value

Usage

```
summary_max(
  col,
  label = NULL,
  digits = NULL,
  prefix = NULL,
  suffix = NULL,
  id = NULL
)
```

Arguments

col	A formula like ~value.
label	Display label for the summary row.
digits	Optional number of digits for formatting (applied in JS).
prefix, suffix	Optional prefix/suffix strings (applied in JS).
id	Optional summary id.

Value

A summary specification object.

summary_mean *Summary: mean value*

Description

Summary: mean value

Usage

```
summary_mean(
  col,
  label = NULL,
  digits = NULL,
  prefix = NULL,
  suffix = NULL,
  id = NULL
)
```

Arguments

col	A formula like ~value.
label	Display label for the summary row.
digits	Optional number of digits for formatting (applied in JS).
prefix, suffix	Optional prefix/suffix strings (applied in JS).
id	Optional summary id.

Value

A summary specification object.

summary_min *Summary: min value*

Description

Summary: min value

Usage

```
summary_min(
  col,
  label = NULL,
  digits = NULL,
  prefix = NULL,
  suffix = NULL,
  id = NULL
)
```

Arguments

col	A formula like ~value.
label	Display label for the summary row.
digits	Optional number of digits for formatting (applied in JS).
prefix, suffix	Optional prefix/suffix strings (applied in JS).
id	Optional summary id.

Value

A summary specification object.

---

summary_sum *Summary: sum of values*

Description

Summary: sum of values

Usage

```
summary_sum(
  col,
  label = NULL,
  digits = NULL,
  prefix = NULL,
  suffix = NULL,
  id = NULL
)
```

Arguments

col	A formula like ~value.
label	Display label for the summary row.
digits	Optional number of digits for formatting (applied in JS).
prefix, suffix	Optional prefix/suffix strings (applied in JS).
id	Optional summary id.

Value

A summary specification object.

tmpl *Create a tooltip/popup template*

Description

Templates use placeholders like `{col}` or `{col:.1f}`. Values are gathered during widget compilation for fast rendering in the browser.

Usage

```
tmpl(template, ..., html = FALSE)
```

Arguments

`template`	A single template string.
`...`	Optional named expressions to bind placeholders (e.g. `mag = ~mag`). If omitted, placeholders are resolved by column name.
`html`	Logical; if `TRUE`, the template is treated as HTML.

Value

A template specification object.

Examples

```
d <- data.frame(
  lon   = runif(1000, -60, 60),
  lat   = runif(1000, -60, 60),
  value = runif(1000, 1, 10)
)
maplamina() |>
  add_circles(d, tooltip = tmpl("Value: {value:.1f}"))
```

transition *Deprecated transition helper*

Description

`transition()` has been removed. Configure animated view switching via `add_views()` using `duration` and `easing`.

Usage

```
transition(...)
```

Arguments

...	Ignored.

Value

This function always errors with `Defunct`.

`view` *Define a view*

Description

A view is a named set of overrides (e.g. radius/opacity) that can be switched via a shared views selector created by `add_views()`.

Usage

```
view(name, ...)
```

Arguments

`name`	View name shown in the UI.
...	Named overrides for layer aesthetics (e.g. `radius = ~mag * 3`, `opacity = 0.3`).

Value

A view specification object.

Examples

```
v <- view("magnitude", radius = ~mag * 3)
v
```

Index