

# Package ‘maplamina’

March 6, 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/jhumble/maplamina>

**BugReports** <https://github.com/jhumble/maplamina/issues>

**Depends** R (>= 4.1.0)

**Imports** htmlwidgets, digest, base64enc, sf

**Suggests** testthat (>= 3.0.0), jsonlite

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Jack Humble [aut, cre]

**Maintainer** Jack Humble <jackhumble.dev@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-03-06 18:10:02 UTC

## Contents

add_circles . . . . .	2
add_filters . . . . .	4
add_fullscreen . . . . .	5
add_geolocate . . . . .	5
add_icons . . . . .	6
add_legend . . . . .	7
add_lines . . . . .	9

add_markers . . . . .	10
add_navigation . . . . .	11
add_panel . . . . .	12
add_polygons . . . . .	13
add_scalebar . . . . .	14
add_summaries . . . . .	15
add_views . . . . .	16
base_tiles . . . . .	17
color_bin . . . . .	18
color_factor . . . . .	19
color_numeric . . . . .	19
color_quantile . . . . .	20
filter_range . . . . .	21
filter_select . . . . .	22
maplamina . . . . .	23
section . . . . .	24
sections . . . . .	25
summary_count . . . . .	25
summary_max . . . . .	26
summary_mean . . . . .	27
summary_min . . . . .	27
summary_sum . . . . .	28
tmpl . . . . .	29
transition . . . . .	29
view . . . . .	30

## Index 31

---

add_circles	<i>Add a circle layer</i>
-------------	---------------------------

---

### Description

Adds a GPU-rendered circle layer (points). Use formulas (e.g.  $\sim\text{mag} * 2$ ) 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 <code>maplamina()</code> .
data	Data for this layer. If NULL, uses the default data supplied to <code>maplamina()</code> .
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. <code>fill_color</code> can be a single color, a vector of colors, or a color scale spec such as <code>color_quantile()</code> .
radius	Circle radius (numeric or formula).
stroke	Logical; draw circle stroke.
tooltip, popup	Optional <code>tmpl()</code> 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 <a href="#">maplamina()</a> .
...	One or more <a href="#">filter_range()</a> / <a href="#">filter_select()</a> 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	<i>Add MapLibre fullscreen control</i>
----------------	--

---

**Description**

Add MapLibre fullscreen control

**Usage**

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

**Arguments**

map	A maplamina widget created by <code>maplamina()</code> .
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	<i>Add MapLibre geolocate control</i>
---------------	---------------------------------------

---

**Description**

Add MapLibre geolocate control

**Usage**

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

**Arguments**

map	A maplamina widget created by <code>maplamina()</code> .
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	<i>Add an icon layer</i>
-----------	--------------------------

---

**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 <a href="#">maplamina()</a> .
data	Data for this layer. If NULL, uses the default data supplied to <a href="#">maplamina()</a> .
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 <code>tmpl()</code> 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 <code>maplamina()</code> .
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	<i>Add a line layer</i>
-----------	-------------------------

---

### 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 <code>maplamina</code> widget created by <code>maplamina()</code> .
data	Data for this layer. Typically an <code>sf</code> object with <code>LINestring</code> geometry.
color, opacity, width	Line color/opacity/width.
tooltip, popup	Optional <code>tmpl()</code> objects (or <code>NULL</code> ) 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 <a href="#">maplamina()</a> .
data	Data for this layer. If NULL, uses the default data supplied to <a href="#">maplamina()</a> .
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	<i>Add MapLibre navigation controls</i>
----------------	---

---

**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	<i>Add a global, presentation-only panel</i>
-----------	--

---

**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 <a href="#">maplamina()</a> .
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 <a href="#">sections(section("id"), ...)</a> .

**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	<i>Add a polygon layer</i>
--------------	----------------------------

---

## 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 <a href="#">maplamina()</a> .
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	<i>Add MapLibre scale bar</i>
--------------	-------------------------------

---

**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 <code>maplamina()</code> .
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	<i>Add summaries to a layer</i>
---------------	---------------------------------

---

**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 <code>maplamina()</code> .
...	One or more <code>summary_*()</code> 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 <code>maplamina()</code> .
...	One or more <code>view()</code> 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	<i>Categorical color scale specification</i>
--------------	--

---

**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	<i>Continuous color scale specification</i>
---------------	---

---

**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 <code>grDevices::hcl.colors()</code> ), or a vector of colors.
domain	Optional numeric range <code>c(min, max)</code> ; 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	<i>Define a range filter</i>
--------------	------------------------------

---

**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	<i>Create a Maplamina map widget</i>
-----------	--------------------------------------

---

**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 <code>add_*</code> () calls when their data argument is NULL.
style	MapLibre style URL (or a named style from <code>base_tiles</code> ).
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 <a href="#">add_views/add_filters</a> , or the component id when bind is omitted).
----	---

**Value**

A panel section object.

**Examples**

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

---

sections	<i>Create a list of panel sections</i>
----------	--

---

**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	<i>Summary: count rows</i>
---------------	----------------------------

---

**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	<i>Summary: max value</i>
-------------	---------------------------

---

**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	<i>Summary: mean value</i>
--------------	----------------------------

---

**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	<i>Summary: min value</i>
-------------	---------------------------

---

**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	<i>Summary: sum of values</i>
-------------	-------------------------------

---

**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	<i>Create a tooltip/popup template</i>
------	--

---

### 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	<i>Deprecated transition helper</i>
------------	-------------------------------------

---

### 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

## \* datasets

base\_tiles, 17

add\_circles, 2

add\_filters, 4

add\_filters(), 21, 22, 24

add\_fullscreen, 5

add\_geolocate, 5

add\_icons, 6

add\_icons(), 10

add\_legend, 7

add\_lines, 9

add\_markers, 10

add\_navigation, 11

add\_panel, 12

add\_polygons, 13

add\_scalebar, 14

add\_summaries, 15

add\_summaries(), 24

add\_views, 16

add\_views(), 24, 29, 30

base\_tiles, 17

color\_bin, 18

color\_factor, 19

color\_factor(), 13

color\_numeric, 19

color\_quantile, 20

color\_quantile(), 3, 13, 14

filter\_range, 21

filter\_range(), 4

filter\_select, 22

filter\_select(), 4

maplamina, 17, 23

maplamina(), 3–6, 8–13, 15, 16

section, 24

section(), 12, 25

sections, 25

sections(), 12

summary\_count, 25

summary\_max, 26

summary\_mean, 27

summary\_min, 27

summary\_sum, 28

tmpl, 29

tmpl(), 3, 7, 9, 11, 14

transition, 29

view, 30

view(), 16