Package: vellumwidget (via r-universe)

July 12, 2026

Title Interactive Web Widgets for Vellum Scenes

Version 0.3.0.9000

Description Turns a 'vellum' scene (or a 'vellumplot' plot) into a self-contained, client-side interactive HTML widget: hover tooltips and highlighting, click selection, rectangular brush-select, pan/zoom, and a toolbar -- driven by the per-element data keys, bounding boxes, and 'scene_model()' metadata that 'vellum' emits. Requires no Shiny and no server round-trip.

License MIT + file LICENSE

URL https://r-vellum.github.io/vellumwidget/, https://github.com/r-vellum/vellumwidget, https://schochastics.r-universe.dev/vellumwidget

BugReports https://github.com/r-vellum/vellumwidget/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.2)

Imports grDevices, htmlwidgets, vellum (>= 0.2.0.9000)

Suggests crosstalk, vellumplot, testthat (>= 3.0.0)

Remotes r-vellum/vellum, r-vellum/vellumplot

Config/testthat/edition 3

RoxygenNote 8.0.0.9000

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libuv1-dev xz-utils libclang-dev

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

Date/Publication 2026-07-12 18:00:35 UTC

RemoteUrl https://github.com/r-vellum/vellumwidget

RemoteRef HEAD

RemoteSha 95240354673912a37ebb42007cff6133f288afa5

Contents

`as_widget` *Turn a vellum scene (or vellumplot plot) into an interactive widget*

Description

`as_widget()` is the terminal verb of the interactivity pipeline: it compiles its input to a `vellum` scene, emits the SVG (with per-element `data-keys`) and the `vellum::scene_model()` element table, and bundles them with the `vellumwidget` JavaScript runtime into a self-contained `htmlwidgets::createWidget()` widget. The result does hover tooltips, hover highlighting, and click selection entirely client-side — no Shiny, no server round-trip. Pan/zoom, brush, and selection work with mouse, touch (drag to pan, two-finger pinch to zoom), and keyboard input: the arrows pan and `+`/`-`/`0` zoom and reset, except that with accessibility on (the default) the arrows move between marks while one is focused — see the `a11y` argument.

Usage

```
as_widget(
  x,
  width = NULL,
  height = NULL,
  tooltip = TRUE,
  hover = TRUE,
  select = TRUE,
  brush = TRUE,
  zoom = TRUE,
  toolbar = TRUE,
  nearest = TRUE,
  a11y = TRUE,
  alt = NULL,
  hover_color = NULL,
  selected_color = NULL,
  dim_opacity = NULL,
  tooltip_style = NULL,
  export_filename = NULL,
  export_scale = NULL,
  group = NULL,
  crosstalk = NULL,
  select_mode = c("multiple", "single"),
  elementId = NULL
)
```

Arguments

x	A vellumplot plot (a PlotSpec / PlotComposition) or a vellum scene — anything vellum::as_vellum_scene() accepts.
width, height	Widget size (any valid CSS size, or NULL to size from the scene). Passed to htmlwidgets::createWidget().
tooltip, hover, select	Toggles for the three hover/click interactions (all TRUE).
brush, zoom, toolbar	Toggles for rectangular brush-select, wheel/drag pan-zoom (via the SVG viewBox), and the on-hover toolbar (all TRUE).
nearest	When TRUE (default), hover snaps to the nearest mark within a small radius when the cursor is not directly over one (helps sparse points).
a11y	Accessibility (default TRUE). Makes the widget a keyboard- and screen-reader-navigable chart: the SVG is labelled as an interactive chart (role="graphics-document"), each mark is a focusable graphics-symbol with a roving tabindex (arrow keys move between marks, Enter/Space select, Escape exits), a polite aria-live region announces the focused/selected mark, and a visually-hidden data table lists every mark for assistive tech. FALSE restores the previous behaviour (no chart semantics, marks not focusable).
alt	Accessible label (alt text) for the chart as a whole. Defaults to the scene's own title/description — which vellumplot sets automatically from the plot title and vellumplot::plot_alt() — so an explicit value is only needed for a raw vellum scene or to override.
hover_color, selected_color	Outline colours for hovered / selected elements (any R or CSS colour), applied widget-wide. hover_color = NULL (default) keeps the plain dim-others hover; selected_color = NULL uses the built-in default. A per-mark hover_color/selected_color declared in vellumplot overrides these for that mark.
dim_opacity	Opacity (0–1) of the non-hovered elements while hovering (default 0.28); NULL keeps the default.
tooltip_style	Optional named list styling the tooltip box: background / color (any R or CSS colour), fontsize, and max_width (any CSS length). NULL (default) uses the built-in style. Tooltip text is rendered as safe HTML — an author-built tooltip = (e.g. via glue()) may use <b>/<i>/ for bold/italic/line breaks; data values are escaped and only those inert tags are honoured (no scripts/attributes).
export_filename, export_scale	The download filename base (no extension; default "plot") and the PNG resolution multiplier (default 1) for the toolbar's SVG/PNG export. Exports capture the current (zoomed/panned) view.
group	Optional linking group name. Widgets sharing a group link client-side: selecting (or brushing) in one highlights the same data keys in the others — no Shiny, no crosstalk. Selection projects by hover_group when the marks declare it (select one, select the whole series).
crosstalk	Optional crosstalk::SharedData (or a crosstalk group name string) to link this widget with the crosstalk ecosystem (plotly, leaflet, DT, and crosstalk's filter_*

	inputs). The widget's `data_id`s must match the SharedData's keys. A crosstalk filter hides the non-matching elements (display-tier cross-filter). Requires the crosstalk package.
`select_mode`	`"multiple"` (default; click toggles each element) or `"single"` (click replaces the selection).
`elementId`	Optional explicit widget DOM id.

Details

Interactivity is driven by the keys/metadata a plot declares. In `vellumplot` these come from the reserved `data_id` / `tooltip` / `hover_group` mark arguments; a plot that declares none renders as a static (but still embeddable) SVG. A hovered element with a `data_id` but no `tooltip` shows its key.

The scene metadata `vellumwidget` reads — the `vellum::scene_model()` element table and the SVG `data-key` / `data-vellum-*` attributes — is specified in vellum's "The scene contract" vignette (`vignette("scene-contract", package = "vellum")`).

Value

An htmlwidget of class `"vellumwidget"`.

Examples

```
## Not run:
library(vellumplot)
df <- data.frame(wt = mtcars$wt, mpg = mtcars$mpg, model = rownames(mtcars))
vplot(df) |>
  mark_point(x = wt, y = mpg, tooltip = model, data_id = model) |>
  as_widget()

## End(Not run)
```

`vellumwidget-shiny` *Shiny bindings for vellumwidget widgets*

Description

Standard htmlwidgets output/render helpers so a `vellumwidget` widget can appear in a Shiny app or an interactive R Markdown document.

Usage

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

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

Arguments

`outputId`	Shiny output slot id.
`width, height`	Widget size.
`expr`	An expression producing a `vellumwidget` widget.
`env, quoted`	Standard non-standard-evaluation plumbing.

Value

`vellumwidgetOutput()`: a Shiny output UI element. `renderVellumwidget()`: a Shiny render function.

Reading interactions server-side

A widget rendered as `vellumwidgetOutput("plot")` reports the user's interactions back to the server as reactive inputs, keyed by the output id. All values are the element **data keys** (the `data_id` a `vellumplot` mark declares); map them back to your data by that key.

`input$plot_selected` Character vector of the currently selected keys (click / brush / keyboard, and any selection arriving from a linked widget). Updates as state — re-selecting the same set is a no-op. `character(0)` when nothing is selected.

`input$plot_click` A list `list(key=)` for each click; `key` is `NULL` for a click on empty space. An event input — fires on every click, even the same mark twice.

`input$plot_hover` The hovered key, or `NULL` when the pointer leaves a mark. Updates as state (re-fires only when the hovered key changes).

`input$plot_brush` A list `list(keys=, x0=, y0=, x1=, y1=)` when a brush gesture completes: the selected keys and the brushed rectangle in the scene's device-pixel (viewBox) coordinates. An event input.

These are emitted only inside a live Shiny session; a static render (knitr, pkgdown, `htmltools::save_html()`) produces identical output and no input traffic. Driving the widget *from* the server (setting selection without a re-render) is a planned addition.

See Also

`as_widget()`

Index