Package 'reglScatterplotR'

November 25, 2026

Type Package

Title Interactive WebGL Scatterplots for Biological Data Exploration

Version 1.0.0

Date 2026-10-21

Description Lightweight 'htmlwidgets' interface to the JavaScript 'regl-scatterplot' library, rendering millions of two-dimensional points in the browser via WebGL. Designed for exploratory visualisation of single-cell, spatial transcriptomics and other high-dimensional biological data. Features include synchronised pan/zoom across multiple plots, categorical and continuous colour mapping, lasso selection, range and category filtering, PNG/SVG/PDF export, and Shiny integration. The widget works in the 'RStudio' Viewer, standalone HTML files, Shiny applications and Jupyter notebooks running the 'IRkernel'.

License MIT + file LICENSE

URL https://github.com/george123ya/reglScatterplotR

BugReports https://github.com/george123ya/reglScatterplotR/issues

Encoding UTF-8

biocViews Software, Visualization, SingleCell, Spatial, GeneExpression, Transcriptomics, DimensionReduction, ShinyApps

Depends R (>= 4.6.0)

Imports SingleCellExperiment, SummarizedExperiment, htmlwidgets, jsonlite, viridisLite, RColorBrewer, grDevices, methods, stats, shiny

Suggests testthat (>= 3.0.0), knitr, rmarkdown, markdown, BiocStyle, SpatialExperiment, SeuratObject, anndataR, crosstalk, DT

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

git_url https://git.bioconductor.org/packages/reglScatterplotR

git_branch RELEASE_3_24

git_last_commit 9045094

git_last_commit_date 2026-11-23

Repository Bioconductor 3.24

Date/Publication 2026-11-24

Author George Muñoz [aut, cre] (ORCID:
<https://orcid.org/0009-0007-3765-9967>)

Maintainer George Muñoz <george.munoz@unmsm.edu.pe>

Contents

reglScatterplotR-package 2
enableReglScatterplotSync 3
reglScatterExample 4
reglScatterplot 5
reglScatterplot-shiny 9
toBase64 10
updateReglPointSize 10

Index **12**

`reglScatterplotR-package`
reglScatterplotR: Interactive WebGL Scatterplots for R

Description

Lightweight `htmlwidgets` interface to the JavaScript `regl-scatterplot` library, rendering millions of points in the browser via WebGL. Designed for exploratory visualisation of single-cell, spatial and other high-dimensional biological data, with synchronised pan/zoom across multiple plots, categorical and continuous colour mapping, lasso selection and PNG/SVG/PDF export.

Main functions

- `reglScatterplot()` - create a scatterplot widget.
- `enableReglScatterplotSync()` - link the camera of multiple plots.
- `updateReglPointSize()` - resize points after rendering.

Rendering contexts

The widget renders in RStudio's Viewer pane, in standalone HTML files, inside Shiny apps and inside Jupyter notebooks running the IRkernel (e.g. VSCode's Jupyter extension). Outside of Shiny the JavaScript dependencies are pulled from a CDN on first use.

Author(s)

Maintainer: George Muñoz <george.munoz@unmsm.edu.pe> (ORCID)

Authors:

- George Muñoz <george.munoz@unmsm.edu.pe> (ORCID)

See Also

Useful links:

- https://github.com/george123ya/reglScatterplotR
- Report bugs at https://github.com/george123ya/reglScatterplotR/issues

---

`enableReglScatterplotSync` *Link the camera of multiple scatterplots*

Description

Enables (or disables) synchronised pan/zoom across the `reglScatterplot` widgets identified by `plotIds`. Must be called from inside a Shiny reactive context.

Usage

```
enableReglScatterplotSync(
  plotIds,
  enabled = TRUE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`plotIds`	Character vector of `plotId` strings.
`enabled`	Logical; `TRUE` to enable sync, `FALSE` to disable.
`session`	Shiny session object (defaults to the active one).

Value

Invisibly `NULL`. Called for its side effect of sending a custom message to the browser.

Examples

```
if (interactive()) {
    enableReglScatterplotSync(c("plotA", "plotB"))
}
```

`reglScatterExample` *Example single-cell UMAP embedding*

Description

A small, synthetic single-cell-style dataset for trying out `reglScatterplot()` without downloading anything or installing a Bioconductor data package. It mimics a PBMC UMAP: eight cell-type clusters arranged as Gaussian blobs in two dimensions, plus a handful of continuous covariates and one marker-gene-like gradient.

Usage

```
data(reglScatterExample)
```

Format

A `data.frame` with 3,400 rows (cells) and 7 columns:

UMAP_1, UMAP_2 Numeric. Two-dimensional UMAP coordinates.

celltype Factor with 8 levels (`"CD4 T"`, `"CD8 T"`, `"NK"`, `"B"`, `"Monocyte"`, `"Dendritic"`, `"Platelet"`, `"Progenitor"`). Use as a categorical `colorBy`.

nCount Integer. Total counts per cell (library size).

nFeature Integer. Number of detected features per cell.

percentMito Numeric. Percent mitochondrial counts (0-25).

CD3D Numeric. Log-normalised expression of a T/NK marker, with dropout. Use as a continuous `colorBy`.

Details

The data are simulated (see `data-raw/make_reglScatterExample.R`) and carry no biological meaning beyond being shaped like real scRNA-seq output. They exist purely so the examples, vignette and tests have something realistic to draw.

Source

Simulated. Generation script in `data-raw/`.

Examples

```
data(reglScatterExample)
# Colour by cell type (categorical)
reglScatterplot(reglScatterExample,
    x = "UMAP_1", y = "UMAP_2", colorBy = "celltype"
)
# Colour by a continuous marker gene
reglScatterplot(reglScatterExample,
    x = "UMAP_1", y = "UMAP_2", colorBy = "CD3D",
    continuousPalette = "viridis", vmax = "p99"
)
```

`reglScatterplot` *Interactive WebGL scatterplot*

Description

Renders a `regl-scatterplot` widget capable of displaying millions of points in the browser. Works in standalone HTML, the RStudio Viewer, Shiny applications and Jupyter notebooks (via IRkernel).

Usage

```
reglScatterplot(
  data = NULL,
  x,
  y,
  colorBy = NULL,
  groupBy = NULL,
  assay = NULL,
  filterBy = NULL,
  pointSize = NULL,
  opacity = NULL,
  pointColor = NULL,
  sizeBy = NULL,
  opacityBy = NULL,
  tooltipBy = NULL,
  pixelRatio = NULL,
  categoricalPalette = "Set1",
  continuousPalette = "viridis",
  customPalette = NULL,
  customColors = NULL,
  pointLabels = NULL,
  xlab = "X",
  ylab = "Y",
  title = NULL,
  legendTitle = NULL,
  xrange = NULL,
  yrange = NULL,
  rangePadding = 0.15,
  vmin = NULL,
  vmax = NULL,
  centerZero = FALSE,
  showAxes = TRUE,
  showTooltip = TRUE,
  backgroundColor = NULL,
  axisColor = "#333333",
  legendBg = "#ffffff",
  legendText = "#000000",
  legendOpacity = NULL,
  legendBlur = NULL,
  toolbarPosition = "none",
  zoomOnSelection = FALSE,
```

```
  legendPosition = "top-right",
  draggableLegend = TRUE,
  width = NULL,
  height = NULL,
  enableDownload = FALSE,
  plotId = NULL,
  syncPlots = NULL,
  elementId = NULL,
  dataVersion = NULL,
  masterId = NULL,
  autoFit = FALSE,
  margins = NULL,
  fontSize = 12,
  legendFontSize = 12,
  filteredIndices = NULL,
  selectedIndices = NULL,
  syncState = TRUE
)
```

Arguments

data	Optional input object. Accepts a data.frame (or anything that supports [[), a numeric coordinate matrix, a SingleCellExperiment, a SpatialExperiment, a Seurat object, or an in-memory AnnData (from the anndataR or anndata packages). For a data.frame, x, y, colorBy, groupBy and the columns named in filterBy are looked up by name. For a matrix, the first two columns are used as coordinates unless x / y give column indices or names. When NULL, these arguments must be vectors of matching length. For single-cell objects, see the *Single-cell data structures* section below.
x, y	Either column names (when data is non-NULL) or numeric vectors giving point coordinates. For SingleCellExperiment / SpatialExperiment inputs, pass the reducedDim name as x (e.g. "UMAP"); y is ignored. Use "spatial" for tissue coordinates on a SpatialExperiment.
colorBy	Optional column name or vector used to colour points. Character/factor input is treated as categorical, numeric as continuous.
groupBy	Optional column name or vector. Used by the legend to expose a second categorical filter that intersects with colorBy.
assay	Name of the assay to read when colorBy names a feature row of an SCE (default "logcounts", falls back to assay #1). Ignored for plain data frames.
filterBy	Optional data.frame of additional numeric covariates that the JavaScript widget exposes for range filtering.
pointSize, opacity	Numeric. When NULL, sensible defaults are picked based on the number of points (pointSize = 3 and opacity = 0.8 for small data; pointSize = 1, opacity = 1 once n > 500000).
pointColor	Optional fixed hex colour. When given, overrides colorBy.
sizeBy, opacityBy	Optional numeric column name or vector mapped to point size / opacity. They share one data channel, so both encode the same variable.
tooltipBy	Optional extra fields to show on hover: column name(s) (when data is a data.frame) or a data.frame / named list of vectors.

`pixelRatio` Optional numeric device-pixel-ratio for the WebGL backing store. When `NULL` (default) the widget renders at `max(devicePixelRatio, 2)` for crisp output - this matters in the RStudio Viewer, an embedded browser that reports `devicePixelRatio = 1` even on HiDPI screens, making the plot look soft at the library default. For very large data (`n > 500000`, performance mode) the true ratio is used to keep the pixel count down. Pass an explicit value (e.g. `1` to save GPU memory, `3` for extra-sharp export-quality rendering) to override.

`categoricalPalette`, `continuousPalette`
Names of fallback palettes (a `RColorBrewer` name and a `viridisLite` name respectively).

`customPalette`, `customColors`
Optional explicit palettes. `customColors` is a named character vector mapping level -> hex; `customPalette` is an unnamed vector used in level order.

`pointLabels` Optional character vector aligned to the points; shown in the hover tooltip. Use this for gene names, cell barcodes, etc.

`xlab`, `ylab`, `title`, `legendTitle`
Axis/labels.

`xrange`, `yrange` Optional length-2 numeric vectors fixing the axes instead of using the data range. When supplied, they bypass the automatic padding from `rangePadding`.

`rangePadding` Fractional padding added to each side of the data range when `xrange` / `yrange` aren't supplied (default `0.1` = 10% on each side, matching the `ggplot2` convention). Set to `0` to draw exactly the data range.

`vmin`, `vmax` Continuous-colour clipping. Accepts a number, `"min"`, `"max"`, or a percentile string like `"p99"`.

`centerZero` Logical. If `TRUE`, the continuous colour scale is forced to be symmetric around zero (useful for log-fold-change displays).

`showAxes`, `showTooltip`
Logical toggles.

`backgroundColor`, `axisColor`, `legendBg`, `legendText`
Hex colour overrides for cosmetic styling.

`legendOpacity` Opacity (0-1) of the frosted legend card background. `NULL` (default) uses `0.55`.

`legendBlur` Backdrop blur in pixels behind the legend card. `NULL` (default) uses `10`; set `0` to disable the frosted effect.

`toolbarPosition`
Show an in-plot toolbar (pan / lasso / zoom-to- selection / reset / screenshot). One of `"none"` (default), `"left"` (vertical) or `"top"` (horizontal).

`zoomOnSelection`
Logical; when `TRUE`, the camera zooms to fit the lasso-selected points.

`legendPosition` Starting position of the legend. One of `"top-right"` (default), `"top-left"`, `"bottom-right"`, `"bottom-left"`, or a length-2 numeric vector `c(x, y)` giving absolute pixel offsets from the top-left of the plot.

`draggableLegend`
Logical; when `TRUE` (default), the legend can be dragged with the mouse. Set to `FALSE` to pin it to `legendPosition`.

`width`, `height` Widget dimensions (any valid CSS unit). When `NULL`, `htmlwidgets` picks defaults appropriate to the host context.

enableDownload — Logical. Show the download (PNG/SVG/PDF) button. Defaults to FALSE because the button is unreliable inside IDE iframes (RStudio Viewer, VSCode Jupyter), where browser download dialogs and html2canvas/jsPDF script injections often fail. Set to TRUE explicitly for standalone HTML files or Shiny apps; even then the button is hidden automatically when the widget is running inside an IDE iframe.

plotId — Optional string identifying the plot. Required when linking plots together via enableReglScatterplotSync().

syncPlots — Optional character vector of plot ids to sync with.

elementId — DOM id for the containing div (passed to htmlwidgets).

dataVersion — Optional integer/string used by the JS layer to detect when the underlying data has changed.

masterId — Optional id of another plot whose camera should seed this one's view (used when a plot is added to an already-synced group).

autoFit — Logical. When TRUE, the camera zooms to fit the data region exactly on initial render.

margins — Optional list(top, right, bottom, left) of pixel margins.

fontSize, legendFontSize — Numeric font sizes in pixels.

filteredIndices, selectedIndices — Optional 0-based integer vectors for server-driven filtering / selection.

syncState — Logical. When FALSE, the widget starts with global sync disabled.

Value

An htmlwidgets object of class "reglScatterplot".

Single-cell data structures

When data is a SingleCellExperiment or SpatialExperiment, the function reroutes through a helper that pulls coordinates from reducedDim() (e.g. x = "UMAP") - or from spatialCoords() when x = "spatial" for a SpatialExperiment. colorBy and groupBy may then name either a colData column or a feature in rownames(sce) (in the latter case assay() is read - "logcounts" by default, or the assay named by the assay argument).

When data is a Seurat object, coordinates come from Embeddings() for the reduction named by x (default "UMAP", matched case-insensitively; falls back to the first of umap/tsne/pca present). colorBy and groupBy are resolved with FetchData(), so they may name either a meta.data column or a feature. For Seurat, the assay argument selects the *layer* to read for features ("data", i.e. log-normalised, by default; works with both v4 slots and v5 layers); to colour by a non-default modality such as ADT, set Seurat::DefaultAssay() before calling.

When data is an in-memory AnnData (from anndataR or the anndata CRAN package), coordinates come from an obsm embedding named by x (default "UMAP", auto-prefixed to the scanpy-style "X_umap" and matched case-insensitively). colorBy / groupBy resolve against obs columns or var_names features, and assay selects the layer to read for features (NULL -> X, a layer name, or "raw"). An AnnData read with zellkonverter::readH5AD() arrives as a SingleCellExperiment and uses the SCE path above instead.

See Also

enableReglScatterplotSync(), updateReglPointSize()

Examples

```
set.seed(1L)
df <- data.frame(
    x = rnorm(2000),
    y = rnorm(2000),
    group = sample(letters[1:4], 2000, replace = TRUE),
    score = runif(2000)
)
reglScatterplot(df, x = "x", y = "y", colorBy = "group")
reglScatterplot(df,
    x = "x", y = "y", colorBy = "score",
    continuousPalette = "magma"
)


# Single-cell UMAP from a SingleCellExperiment
# reglScatterplot(sce, dimred = "UMAP", colorBy = "cluster")
# Colour by a gene
# reglScatterplot(sce, dimred = "UMAP", colorBy = "MS4A1")
# Spatial coordinates from a SpatialExperiment
# reglScatterplot(spe, dimred = "spatial", colorBy = "celltype")
```

---

reglScatterplot-shiny *Shiny bindings for reglScatterplot*

Description

Output and render functions for using `reglScatterplot()` inside Shiny applications and interactive R Markdown documents.

Usage

```
reglScatterplotOutput(outputId, width = "100%", height = "600px")

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

Arguments

`outputId`	Output variable name (character).
`width, height`	CSS unit (e.g. `"100%"`, `"600px"`).
`expr`	An expression that produces a `reglScatterplot` widget.
`env`	Environment in which to evaluate `expr`.
`quoted`	Is `expr` already quoted? Default `FALSE`.

Value

`reglScatterplotOutput()` returns a Shiny output element; `renderReglScatterplot()` returns a render function.

Examples

```
if (interactive()) {
    ui <- shiny::fluidPage(reglScatterplotOutput("plot"))
    server <- function(input, output, session) {
        output$plot <- renderReglScatterplot({
            reglScatterplot(iris,
                x = "Sepal.Length", y = "Sepal.Width",
                colorBy = "Species"
            )
        })
    }
    shiny::shinyApp(ui, server)
}
```

---

toBase64 *Encode a numeric vector as a base64 Float32 payload*

Description

Serialises a numeric vector as little-endian 32-bit floats and prefixes the result with `"base64:"` so the companion JavaScript widget recognises it as a binary buffer rather than a JSON array. Using this transfer path keeps the payload size at ~25% of the equivalent JSON representation.

Usage

```
toBase64(vec)
```

Arguments

`vec`	Numeric vector. `NULL` is returned unchanged.

Value

Character string of the form `"base64:..."` or `NULL`.

Examples

```
toBase64(c(1, 2, 3.5))
```

---

updateReglPointSize *Update the point size of one or more scatterplots*

Description

Update the point size of one or more scatterplots

Usage

```
updateReglPointSize(plotIds, size, session = shiny::getDefaultReactiveDomain())
```

Arguments

plotIds	Character vector of plot ids whose size should be updated.
size	Numeric pixel size.
session	Shiny session.

Value

Invisibly NULL.

Examples

```
if (interactive()) {
    updateReglPointSize("plotA", size = 5)
}
```

Index

∗ **datasets**
reglScatterExample, 4
∗ **internal**
reglScatterplotR-package, 2

enableReglScatterplotSync, 3
enableReglScatterplotSync(), *2*, *8*

reglScatterExample, 4
reglScatterplot, 5
reglScatterplot(), *2*, *4*
reglScatterplot-shiny, 9
reglScatterplotOutput
*(*reglScatterplot-shiny*)*, 9
reglScatterplotR
*(*reglScatterplotR-package*)*, 2
reglScatterplotR-package, 2
renderReglScatterplot
*(*reglScatterplot-shiny*)*, 9

toBase64, 10

updateReglPointSize, 10
updateReglPointSize(), *2*, *8*