Package 'canvasXpress'

September 29, 2026

Version 1.70.3

Title Visualization Package for CanvasXpress in R

Description Enables creation of visualizations using the CanvasXpress framework in R. CanvasXpress is a standalone JavaScript library for reproducible research with complete tracking of data and end-user modifications stored in a single PNG image that can be played back. See <https://www.canvasxpress.org> for more information.

Type Package

License GPL-3

Encoding UTF-8

Language en-US

URL https://github.com/neuhausi/canvasXpress

BugReports https://github.com/neuhausi/canvasXpress/issues

Depends R (>= 3.6)

Imports htmlwidgets (>= 1.0), htmltools, httr, jsonlite, stats, utils

RoxygenNote 7.3.3

Suggests repr, scales, ggalluvial, shiny (>= 1.1.0), canvasXpress.data, dplyr, DT, glue, grid, knitr, png, readr, rlang, rmarkdown, stringr, testthat, tibble, tidyr, limma, ggplot2, survminer, S7, patchwork, GGally, ggpubr, ggpattern, ggh4x

VignetteBuilder knitr

NeedsCompilation no

Author Isaac Neuhaus [aut], Connie Brett [aut, cre]

Maintainer Connie Brett <connie@aggregate-genius.com>

Repository CRAN

Date/Publication 2026-09-29 10:50:02 UTC

Contents

canvasXpress-package . . . 2
canvasXpress . . . 3
canvasXpress.json . . . 4
canvasXpressOutput . . . 5
cxConfigParams . . . 6
cxHtmlPage . . . 7
cxShinyExample . . . 8
cxValidateConfig . . . 8
ggplot.as.list . . . 9
ggplot.decompiled . . . 10
renderCanvasXpress . . . 11

Index **12**

`canvasXpress-package` *CanvasXpress Visualization Package*

Description

A package to assist in creating visualizations in CanvasXpress in R.

Details

CanvasXpress is a standalone JavaScript library for reproducible research with complete tracking of data and end-user modifications stored in a single PNG image that can be played back for an extensive set of visualizations.

More Information

https://www.canvasxpress.org

```
browseVignettes(package = "canvasXpress")
```

Author(s)

Maintainer: Connie Brett <connie@aggregate-genius.com>

Authors:

- Isaac Neuhaus <imnphd@gmail.com>

See Also

Useful links:

- https://github.com/neuhausi/canvasXpress
- Report bugs at https://github.com/neuhausi/canvasXpress/issues

canvasXpress *HTML Widget Creation*

Description

Custom HTML widget creation function based on widget YAML and JavaScript for use in any html-compatible context

Usage

```
canvasXpress(
  data = NULL,
  smpAnnot = NULL,
  varAnnot = NULL,
  graphType = "Scatter2D",
  events = NULL,
  afterRender = NULL,
  pretty = FALSE,
  digits = 4,
  width = 600,
  height = 400,
  destroy = FALSE,
  validate = FALSE,
  ...
)
```

Arguments

data	data.frame-, matrix-, list- , or ggplot- classed object
smpAnnot	additional data that applies to samples (columns)
varAnnot	additional data that applies to variables (rows)
graphType	type of graph to be plotted - default = "Scatter2D"
events	user-defined events (e.g. mousemove, mouseout, click and dblclick)
afterRender	event triggered after rendering
pretty	print tagged code (JSON/HTML) nicely - default = FALSE
digits	display digits - default = 4
width	plot width (valid CSS units) - default = 600px
height	plot height (valid CSS units) - default = 400px
destroy	used to indicate removal of a plot - default = FALSE
validate	if TRUE, check the config parameters against the CanvasXpress parameter catalog and warn on unknown parameters or invalid enumerated values (see cxValidateConfig) - default = FALSE
...	additional parameters passed to canvasXpress

Value

htmlwidgets object

Piping Support

Piping is supported (both the magrittr ' canvasXpress object into data parameter. Any new parameters will be added to the original configuration of the object, any parameters with data that existed before will be replaced, and any parameters set to null will be removed. It is important to note that primary data changes are not allowed in this construct - which means that anything specified by using the data, varAnnot, or smpAnnot parameters cannot be changed from the original values.

---

canvasXpress.json *HTML Widget Creation using JSON input*

Description

Custom HTML widget creation function based on widget YAML and JavaScript for use in any html-compatible context using raw JSON input. Validation of data and configuration is deferred completely to the canvasXpress JavaScript library.

Usage

```
canvasXpress.json(
  json,
  pretty = FALSE,
  digits = 4,
  width = 600,
  height = 400,
  destroy = FALSE
)
```

Arguments

`json`	JSON string or object
`pretty`	print tagged code (JSON/HTML) nicely - default = FALSE
`digits`	display digits - default = 4
`width`	plot width (valid CSS units) - default = 600px
`height`	plot height (valid CSS units) - default = 400px
`destroy`	used to indicate removal of a plot - default = FALSE

Details

For the formatting of the JSON input object see

Note: this function is intended for use by advanced users who are experimenting with or need to utilize the json-formatted input to canvasXpress and are comfortable debugging chart issues in a browser (JavaScript) context instead of in R.

Value

htmlwidgets object

More Information

https://www.canvasxpress.org

Examples

```
my_json <- '{ "data": {"y": { "vars": ["Performance"],
                              "smps": ["January"],
                              "data": [[85]] }},
              "config": { "graphType": "Meter",
                          "meterType": "gauge" }}'

canvasXpress.json(my_json)
```

---

canvasXpressOutput *Shiny UI function*

Description

Output creation function for canvasXpressOutput in Shiny applications and interactive Rmd documents

Usage

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

Arguments

outputId	shiny unique ID
width	width of the element - default = 100%
height	height of the element - default = 400px

Value

Output function that enables the use of the widget in applications

See Also

renderCanvasXpress

cxShinyExample

cxConfigParams *CanvasXpress configuration parameter catalog*

Description

Returns a data.frame describing every CanvasXpress config parameter - its name, type, default value, allowed values (for enumerated parameters) and a one-line description. The catalog is generated from the CanvasXpress config schema (the same source as the JavaScript `.d.ts` and Python `CXConfig` types) and shipped with the package under `inst/config/config-params.json`.

Usage

```
cxConfigParams()
```

Details

Because `canvasXpress` accepts config parameters through `...`, R cannot autocomplete them; this catalog is the R equivalent of those typed surfaces - a searchable, documented reference you can query programmatically and validate against with `cxValidateConfig`.

Value

A data.frame with columns `parameter`, `type`, `default`, `options` (a list-column of allowed values, or `NULL`) and `description`. The result is memoised for the session.

See Also

`cxValidateConfig`, `canvasXpress`

Examples

```
## Not run:
params <- cxConfigParams()
# look up one parameter
params[params$parameter == "graphType", ]
# every parameter whose description mentions "legend"
params[grepl("legend", params$description, ignore.case = TRUE), "parameter"]

## End(Not run)
```

cxHtmlPage *Stand-Alone HTML Page Creation*

Description

This function creates and returns a stand-alone HTML page containing the given canvasXpress object. Width and height can be inferred from the canvasXpress object (default) or overridden for the page output.

Usage

```
cxHtmlPage(chartObject, width = NULL, height = NULL)
```

Arguments

chartObject	a canvasXpress plot object
width	plot width override for the HTML page (valid CSS units) - default = NULL
height	plot height override for the HTML page (valid CSS units) - default = NULL

Value

a character string containing a self-contained html page

Examples

```
## Not run:
my_chart <- canvasXpress(data      = data.frame(Sample1 = c(33, 48),
                                                Sample2 = c(44, 59),
                                                Sample3 = c(55, 6)),
                         graphType = "Bar",
                         title     = "Example Bar Chart",
                         width     = "600px")

# create a page using the chart dimensions on my_chart
html_page <- cxHtmlPage(my_chart)

# or change the chart width/height for this page:
html_page <- cxHtmlPage(my_chart, width = "100%", height = "70vh")

# save page for viewing/sharing
writeLines(html_page, tempfile(fileext = ".html"))

## End(Not run)
```

cxShinyExample *Create Shiny Example Application*

Description

This function runs one of the available shiny example applications. To see the list of available example applications run the function with no inputs

Usage

```
cxShinyExample(example = NULL)
```

Arguments

example	character name of a valid example application.

Value

Launches a running shiny example application

See Also

canvasXpressOutput

renderCanvasXpress

cxValidateConfig *Validate a CanvasXpress configuration against the parameter catalog*

Description

Checks a named config list against `cxConfigParams` and reports parameters that are not recognised and enumerated parameters set to a value outside their allowed options. This mirrors the compile-time checking the JavaScript `.d.ts` and Python `CXConfig` types give consumers, for R code where config flows through `...` and is otherwise unchecked.

Usage

```
cxValidateConfig(config, strict = FALSE)
```

Arguments

config	A named list of CanvasXpress config parameters (the same `name = value` pairs you would pass to `canvasXpress` via `...`).
strict	Logical; if `TRUE`, raise an error when any issue is found instead of warning. Default `FALSE`.

Details

Recognised-but-unlisted parameters (obfuscation aliases, or a parameter newer than the installed catalog) are reported as unknown; that is a hint, not proof of an error - canvasXpress still accepts any parameter.

Value

Invisibly, a list with elements unknown (character vector of unrecognised parameter names) and bad_options (a named list mapping each offending enumerated parameter to the invalid value supplied). A config with no issues returns two empty elements and emits no message.

See Also

cxConfigParams, canvasXpress

Examples

```
## Not run:
cxValidateConfig(list(graphType = "Bar", colorScheme = "Tableau"))   # clean
cxValidateConfig(list(graphType = "NotAType", wat = 1))              # warns

## End(Not run)
```

---

ggplot.as.list *Converts a ggplot object to a list that can be used by CanvasXpress.*

Description

Converts a ggplot object to a list that can be used by CanvasXpress.

Usage

```
ggplot.as.list(o, ...)
```

Arguments

o	the ggplot object
...	additional parameters to the function

`ggplot.decompiled` *Reconstruct approximate ggplot2 source code from a ggplot object.*

Description

Walks a ggplot object and returns code that renders an equivalent plot. This is BEST-EFFORT: functions passed to stats (e.g. `fun = mean`) round-trip only by name, custom themes collapse to a note (a complete theme carries every element, so the original `theme_*()`/`element_*()` calls cannot be recovered), and the data pipeline that built `o$data` is unrecoverable.

Usage

```
ggplot.decompiled(o, data_name = "dat")
```

Arguments

`o`	a ggplot object.
`data_name`	the symbol used for the data argument in the emitted `ggplot(...)` call (default `"dat"`).

Details

The result is also attached to the output of `ggplot.as.list` as the `decompiled` field.

Value

A single character string containing the reconstructed code, with one ggplot layer/scale/label per line. Use `cat()` or `writeLines()` to print it. Returns `NA_character_` for non-ggplot input.

Examples

```
## Not run:
  p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) + ggplot2::geom_point()
  cat(ggplot.decompiled(p, "mtcars"))

## End(Not run)
```

`renderCanvasXpress` *Shiny Render function*

Description

Render function for canvasXpressOutput in Shiny applications and interactive Rmd documents

Usage

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

Arguments

`expr`	expression used to render the canvasXpressOutput
`env`	environment to use - default = parent.frame()
`quoted`	whether the expression is quoted - default = FALSE

Value

Render function that enables the use of the widget in applications

Destroy

When there exists a need to visually remove a plot from a Shiny application when it is not being immediately replaced with a new plot use the destroy option as in:

```
renderCanvasXpress({canvasXpress(destroy = TRUE)})
```

See Also

canvasXpressOutput

cxShinyExample

Index

canvasXpress, 3, *6*, *8, 9*
canvasXpress-package, 2
canvasXpress.json, 4
canvasXpressOutput, 5, *8*, *11*
cxConfigParams, 6, *8, 9*
cxHtmlPage, 7
cxShinyExample, *5*, 8, *11*
cxValidateConfig, *3*, *6*, 8

ggplot.as.list, 9, *10*
ggplot.decompiled, 10

renderCanvasXpress, *5*, *8*, 11