

Package: maidr (via r-universe)

October 21, 2026

Type Package

Title Multimodal Access and Interactive Data Representation

Version 0.5.0.9000

URL <https://github.com/xability/r-maidr>, <https://r.maidr.ai/>

BugReports <https://github.com/xability/r-maidr/issues>

Description Provides accessible, interactive visualizations through the 'MAIDR' (Multimodal Access and Interactive Data Representation) system. Converts 'ggplot2' and Base R plots, and experimentally 'lattice' charts, into accessible HTML/SVG formats with keyboard navigation, screen reader support, and 'sonification' capabilities. Supports bar charts (simple, grouped, stacked), pie charts, histograms, line plots, step plots, scatter plots, box plots, violin plots, candlestick (OHLC) charts, heat maps, density/smooth curves, faceted plots, multi-panel layouts (including patchwork), and multi-layered plot combinations. Also makes 'plotly', 'highcharter' and 'echarts4r' 'htmlwidgets' accessible by attaching the matching 'MAIDR' JavaScript adapter. Enables data exploration for users with visual impairments through multiple sensory modalities. For more details see the 'MAIDR' project <<https://maidr.ai/>>.

License GPL (>= 3)

Encoding UTF-8

Language en

Depends R (>= 4.0.0)

Imports base64enc, curl, ggplot2, ggplotify, grid, gridGraphics, htmltools, htmlwidgets, jsonlite, methods, shiny, stats, svglite (>= 2.1.1), xml2, rlang, R6, Rcpp

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), lintr, styler, goodpractice, cyclocomp, knitr (>= 1.44), evaluate, xfun (>= 0.43), rmarkdown, quarto, patchwork, scales, tibble, tidyquant, quantmod, TTR, zoo, xts, pkgload, hexbin, vioplot, wordcloud, sm, pROC, ROCR, survival,

survminer, yardstick, dplyr, ggdist, ggraph, igraph, Hmisc,
 plotly, highcharter, echarts4r, lattice, latticeExtra,
 pheatmap, ComplexHeatmap, withr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-
 dev libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

Repository <https://xability.r-universe.dev>

Date/Publication 2026-10-21 15:30:40 UTC

RemoteUrl <https://github.com/xability/r-maidr>

RemoteRef HEAD

RemoteSha 0a1dc4b0521b6357f236fdc067ff0f70bbc7670f

Contents

base-r-wrappers	3
combine_facet_layer_data	7
combine_facet_layer_selectors	7
extract_leaf_plot_layout	8
extract_patchwork_leaves	8
find_children_by_type	9
find_graphics_plot_grob	9
find_patchwork_panels	10
generate_robust_css_selector	10
generate_robust_selector	11
get_facet_groups	11
maidr-options	12
maidr_download_dotpad_sdk	15
maidr_gantt	17
maidr_get_fallback	19
maidr_htmlwidget	20
maidr_manhattan	22
maidr_off	24
maidr_on	25
maidr_output	28
maidr_pr_curve	28
maidr_roc	31
maidr_set_fallback	33
maidr_survival	34
maidr_volcano	37
map_visual_to_dom_panel	39
organize_facet_grid	40

pheatmap	40
process_facet_panel	41
process_faceted_plot_data	42
process_patchwork_panel	43
process_patchwork_plot_data	44
render_maird	44
run_example	46
save_html	47
show	50
Index	55

base-r-wrappers	<i>Functions maidr masks on attach</i>
-----------------	--

Description

library(maidr) puts maidr’s own copies of the Base R plotting functions ahead of the originals on the search path, and R says so with its "The following objects are masked" notice. Each copy is a wrapper: it records the call so that `show()` and `save_html()` can render an accessible chart, then calls the original and returns what the original returns. With interception off (`maidr_off()`), or `options(maidr.base_r = FALSE)`) the wrappers pass straight through.

Usage

- barplot(...)
- plot(...)
- hist(...)
- boxplot(...)
- image(...)
- heatmap(...)
- contour(...)
- matplot(...)
- curve(...)
- dotchart(...)
- stripchart(...)
- stem(...)

```
pie(...)  
mosaicplot(...)  
assocplot(...)  
pairs(...)  
coplot(...)  
persp(...)  
sunflowerplot(...)  
fourfoldplot(...)  
spineplot(...)  
cdplot(...)  
qqnorm(...)  
qqplot(...)  
qqline(...)  
filled.contour(...)  
acf(...)  
pacf(...)  
ccf(...)  
cpgram(...)  
spectrum(...)  
monthplot(...)  
termplot(...)  
lag.plot(...)  
biplot(...)  
interaction.plot(...)
```

```
bxp(...)
stars(...)
vioplot(...)
wordcloud(...)
chartSeries(...)
lines(...)
points(...)
text(...)
mtext(...)
abline(...)
segments(...)
arrows(...)
polygon(...)
rect(...)
symbols(...)
legend(...)
axis(side, at = NULL, labels = TRUE, ...)
title(...)
grid(...)
par(...)
layout(...)

## S3 method for class 'screen'
split(...)
```

Arguments

... Arguments passed to the original graphics function.

side, at, labels `axis()`'s own arguments, which its wrapper names so that the tick labels a chart is given can be recorded; passed on to `graphics::axis()` unchanged.

Details

What is masked:

From `graphics`: the high-level plotting functions `barplot()`, `plot()`, `hist()`, `boxplot()`, `image()`, `contour()`, `matplot()`, `curve()`, `dotchart()`, `stripchart()`, `stem()`, `pie()`, `mosaicplot()`, `assocplot()`, `pairs()`, `coplot()`, `persp()`, `sunflowerplot()`, `fourfoldplot()`, `spineplot()`, `cdplot()`, `filled.contour()`, `bxp()` and `stars()`; the low-level additions `lines()`, `points()`, `text()`, `mtext()`, `abline()`, `segments()`, `arrows()`, `polygon()`, `rect()`, `symbols()`, `legend()`, `axis()`, `title()` and `grid()`; and the layout functions `par()`, `layout()` and `split.screen()`.

From `stats`: `heatmap()`, `qqnorm()`, `qqplot()`, `qqline()`, `acf()`, `pacf()`, `ccf()`, `cpgram()`, `spectrum()`, `monthplot()`, `termplot()`, `lag.plot()`, `biplot()` and `interaction.plot()`.

`plot()` is also masked from `base`, where its generic has lived since R 4.0, and `show()` from `methods`: the S4 display generic, which `maidr`'s `show()` hands back any object that is not a plot.

`vioplot::vioplot()`, `wordcloud::wordcloud()` and `quantmod::chartSeries()` are wrapped as well, once their package is loaded. So is `pheatmap::pheatmap()`, whose wrapper keeps the matrix it draws rather than recording a call; see `pheatmap()`.

`show()` and `methods::show()`:

`maidr`'s `show()` takes a `ggplot2` or `lattice` object or, with no argument, the last recorded Base R chart. Anything else it is given goes to `methods::show()`, so `show(x)` on an S4 object prints as it did before `maidr` was attached. In a script or a package, where what is masked depends on what else is attached, call `maidr::show()` and `methods::show()` by name.

Attach order for `vioplot`, `wordcloud` and `quantmod`:

These three are wrapped into `maidr`'s namespace when their package loads, so a bare call reaches the wrapper only while `package:maidr` sits ahead of the package on the search path. Attach them *before* `maidr`:

```
library(vioplot)
library(maidr)
```

Attached after it, the package masks the wrapper, a bare `vioplot()`, `wordcloud()` or `chartSeries()` draws without being recorded, and `show()` reports that no Base R plot was detected. `maidr` says so at the moment the package is attached and again in that error. The other way round it is `maidr::vioplot()`, `maidr::wordcloud()` or `maidr::chartSeries()`, called explicitly.

Attach order for `ROCR`:

`ROCR` exports an S4 generic for `plot()`, so that `plot()` of a performance object draws its curve. Attached after `maidr`, that generic masks `maidr`'s `plot()`, and no bare `plot()` call is recorded, whatever it draws. Attach `ROCR` before `maidr`, or call `maidr::plot()`; `maidr` says so when `ROCR` is attached after it. A precision-recall curve drawn as `plot(performance(pred, "prec", "rec"))` is then read as one, with the cutoff of each point.

Calling an original directly:

The wrappers add nothing to the drawing and return what the original returns, so there is rarely a reason to go around them. `graphics::barplot()` does, and draws a chart `maidr` does not record.

Value

Same as the original Base R function (invisibly when applicable).

See Also

`show()` and `save_html()`; `maidr_on()` and `maidr_off()` for turning interception on and off; `?"maidr-options"`.

`combine_facet_layer_data`*Combine data from multiple layers in facet processing*

Description

Combine data from multiple layers in facet processing

Usage

```
combine_facet_layer_data(layer_results)
```

Arguments

`layer_results` List of layer processing results

Value

Combined data

`combine_facet_layer_selectors`*Combine selectors from multiple layers in facet processing*

Description

Combine selectors from multiple layers in facet processing

Usage

```
combine_facet_layer_selectors(layer_results)
```

Arguments

`layer_results` List of layer processing results

Value

Combined selectors

`extract_leaf_plot_layout`*Extract layout from a single leaf ggplot*

Description

Mirrors the single-plot path in `Ggplot2PlotOrchestrator`: labels are read from the BUILT plot first. `ggplot2` v4 resolves derived labels only while building – the stat-computed "count" of `geom_bar()`, and the mapped column names – so an unbuilt labels holds nothing but the explicit `labs()` overrides.

Usage

```
extract_leaf_plot_layout(leaf_plot, leaf_built = NULL)
```

Arguments

<code>leaf_plot</code>	The ggplot object
<code>leaf_built</code>	The leaf's built plot, or NULL when it is unavailable

Value

Layout with title and axes

`extract_patchwork_leaves`*Recursively extract leaf ggplots in patchwork addition order*

Description

The order matches panel discovery order in `find_patchwork_panels()` (patches first, then the plot carried by the patchwork object itself), which is how leaves are paired with panels.

Usage

```
extract_patchwork_leaves(node)
```

Arguments

<code>node</code>	Patchwork node or ggplot object
-------------------	---------------------------------

Value

List of leaf ggplot objects

find_children_by_type *Find children matching a type pattern*

Description

Find children matching a type pattern

Usage

```
find_children_by_type(parent_grob, pattern)
```

Arguments

parent_grob	The parent grob to search
pattern	The pattern to match in grob names

Value

Vector of matching child names

find_graphics_plot_grob
Find grob by element type pattern

Description

Searches recursively through a grob tree to find a grob whose name matches the pattern: graphics-plot-<number>-<element type>

Usage

```
find_graphics_plot_grob(grob, element_type, plot_index = NULL)
```

Arguments

grob	The grob tree to search (typically from ggplotify::as.grob())
element_type	The element type to search for (e.g., "rect", "lines", "points")
plot_index	Optional plot index to match (for multipanel layouts)

Value

The name of the first matching grob, or NULL if not found

`find_patchwork_panels` *Discover panels via gtable layout rows named `^panel-<num>` or `^panel-<row>-<col>` Returns a `data.frame` with `panel_index`, `name`, `t`, `l`, `row`, `col`*

Description

Discover panels via gtable layout rows named `^panel-<num>` or `^panel-<row>-<col>` Returns a `data.frame` with `panel_index`, `name`, `t`, `l`, `row`, `col`

Usage

```
find_patchwork_panels(gtable)
```

Arguments

`gtable` Gtable object

Value

Data frame with panel information

`generate_robust_css_selector`
Generate robust CSS selector from grob name

Description

Creates a CSS selector that targets SVG elements by their ID pattern, without relying on panel structure or hardcoded values.

Usage

```
generate_robust_css_selector(grob_name, svg_element)
```

Arguments

`grob_name` The name of the grob (e.g., "graphics-plot-1-rect-1")
`svg_element` The SVG element type to target (e.g., "rect", "polyline")

Value

A robust CSS selector string, or NULL if `grob_name` is invalid

`generate_robust_selector`*Generate robust selector for any element type*

Description

Creates a robust CSS selector that works regardless of panel structure. This is the main function that layer processors should use.

Usage

```
generate_robust_selector(  
  grob,  
  element_type,  
  svg_element,  
  plot_index = NULL,  
  max_elements = NULL  
)
```

Arguments

<code>grob</code>	The grob tree to analyze
<code>element_type</code>	The element type to search for (e.g., "rect", "lines")
<code>svg_element</code>	The SVG element to target (e.g., "rect", "polyline")
<code>plot_index</code>	Optional plot index for multipanel layouts
<code>max_elements</code>	Optional limit on number of elements to target

Value

A robust CSS selector string, or NULL if element not found

`get_facet_groups`*Get facet group information for a panel*

Description

Get facet group information for a panel

Usage

```
get_facet_groups(panel_info, built)
```

Arguments

<code>panel_info</code>	Panel information from layout
<code>built</code>	Built plot data

Value

List of facet group information

<code>maidr-options</code>	<i>MAIDR Package Options</i>
----------------------------	------------------------------

Description

Configure MAIDR interception and display behavior using R's options system.

Available Options

`maidr.auto_show` Logical. Master switch for all MAIDR interception. When FALSE, all plotting functions behave as standard R. Default: TRUE.

`maidr.base_r` Logical. Enable Base R plot interception. When TRUE, Base R plots are captured and displayed in the MAIDR viewer, and the ones an R Markdown or Quarto document draws are accessible charts; when FALSE, a document's Base R plots are knitr's static figures (svglite in HTML). Default: TRUE.

`maidr.ggplot2` Logical. Enable ggplot2 auto-display. When TRUE, ggplot2 objects are automatically rendered in the MAIDR viewer instead of the standard graphics device, and the ones an R Markdown or Quarto document draws are accessible charts; when FALSE, a document's ggplot2 plots are knitr's static figures (svglite in HTML). Default: TRUE.

`maidr.lattice` Logical. Enable lattice auto-display. When TRUE, a lattice (trellis) object printed at the console to the screen is rendered in the MAIDR viewer, and the ones an R Markdown or Quarto document draws, returned from a chunk or printed in it, are accessible charts. A print into a file device, or one that places the chart on a shared page with `split`, `position` or `more`, is drawn by lattice as before. Default: TRUE.

`maidr.startup_message` Logical. Show startup message when package is loaded. Default: TRUE.

`maidr.knitr_dev` Logical. In an R Markdown or Quarto document rendered to HTML, `maidr` records the static figures of a chunk with `svglite` rather than knitr's default `png`, so they are vector images like its charts. Only that default is replaced: a device a chunk names (`dev = "png"` in its header or a `#|` line) is kept, as is a document device other than `png`, the device of a cached chunk, of an animation (`fig.show = "animate"`) and of a chunk that crops its figures or hands them to `fig.process`, and that of a chunk in another language than R. The switch starts with the chunk after the one that installs `maidr` into the document: the chunk that loads `maidr`, or, in a session that loaded it before the render, the first one that draws a chart or calls `maidr_on()`. Set this to FALSE, before the render or in the setup chunk, to keep `png` for every chunk: a document whose `png` is a choice, `dev: png` in its YAML header or Quarto's knitr: `opts_chunk: dev: png`, cannot be told from the default. Charts are inline either way. Default: TRUE.

- `maidr.dotpad_sdk_url` Character. URL of a copy of the DotPad tactile-display SDK module (DotPadSDK-3.0.3.js) that you serve yourself. `maidr.js` does not bundle the SDK, whose braille engine is a 14 MB liblouis build; unless told otherwise it imports the vendor's copy from `jsDelivr` the first time a DotPad is connected, from a document rendered with `use_cdn = FALSE` as much as any other. Set this to keep that path off the network too. Falls back to the environment variable `MAIDR_DOTPAD_SDK_URL`. Default: unset.
- `maidr.dotpad_asset_base_url` Character. URL of the directory holding the SDK's braille engine (`liblouis.js`, `.wasm`, `.data`), needed only when it is not the `lib/` folder beside the module. Falls back to the environment variable `MAIDR_DOTPAD_ASSET_BASE_URL`. Default: unset.
- `maidr.dotpad_sdk_dir` Character. Directory where `maidr_download_dotpad_sdk()` writes the SDK and where `show()` and `save_html()` look for it when a document is rendered with `use_cdn = FALSE`. Falls back to the environment variable `MAIDR_DOTPAD_SDK_DIR`, then to a per-user cache directory. Default: unset.
- `maidr.cdn_version` Character. Which MAIDR.js the CDN paths load: a version such as "4.9.0" (a leading v is accepted), "bundled" for the version bundled with this package (`maidr::MAIDR_VERSION`), or "latest" for `jsDelivr`'s @latest tag. Either tag skips the version lookup described below. Anything else warns once and is ignored. Falls back to the environment variable `MAIDR_CDN_VERSION`. Default: unset, which loads the latest published version.
- `maidr.cdn_timeout` Numeric. Seconds allowed for the whole CDN version lookup, clamped to between 0.1 and 30. Falls back to the environment variable `MAIDR_CDN_TIMEOUT`. Default: 3.
- `maidr.locale_base_url` Character, or FALSE. Where `maidr.js` fetches a language other than English from, in the documents that load the bundled `maidr.js` (see the section on languages below): a directory URL to serve the locale packs yourself, or "" or FALSE to declare nothing, which keeps such a document off the network and in English. A URL set here is declared in CDN documents too. Falls back to the environment variable `MAIDR_LOCALE_BASE_URL`. Default: unset, which declares the packs of the bundled version on `jsDelivr`.
- `maidr.hover_mode` Character. The hover mode a chart starts with when it is not given one: "pointermove" (hovering moves the reader's position and its highlight), "click" (only a click moves it) or "off" (the pointer is ignored, and only the keyboard moves it). It is the `hover_mode` of every chart whose `show()`, `save_html()`, `render_maidr()` or `maidr_htmlwidget()` call sets none, and of the charts an R Markdown or Quarto document draws and those printed at the console; a `plotly` widget, which the MAIDR core reads by itself, takes none. It is only where the reader starts: a reader who has changed the Hover Mode setting keeps theirs. Read by `maidr.js` 4.15.0 and later, the version bundled with this package; an older one, which `maidr.cdn_version` can pin, ignores it. Anything else is an error when a chart is drawn. Default: unset, which writes nothing, so `maidr.js` uses the reader's setting, "pointermove" unless they changed it.

Setting Options

Options can be set in your `.Rprofile` to persist across sessions:

```
# Disable ggplot2 interception by default
options(maidr.ggplot2 = FALSE)

# Draw printed lattice charts with lattice, as without maidr
```

```

options(maidr.lattice = FALSE)

# Disable all interception
options(maidr.auto_show = FALSE)

# Suppress startup message
options(maidr.startup_message = FALSE)

# Keep knitr's png for the static figures of an R Markdown or Quarto page
options(maidr.knitr_dev = FALSE)

# Serve the DotPad SDK from your own server instead of jsDelivr
options(
  maidr.dotpad_sdk_url = "https://example.org/vendor/DotPadSDK-3.0.3.js",
  maidr.dotpad_asset_base_url = "https://example.org/vendor/lib/"
)

# Serve MAIDR's language packs yourself, or never fetch one (English only)
options(maidr.locale_base_url = "https://example.org/maidr/")
options(maidr.locale_base_url = FALSE)

# Move the reader through a chart only on a click, not on hover
options(maidr.hover_mode = "click")

```

DotPad SDK and offline documents

The two `maidr.dotpad*_url` options are written into every document this package produces (`show()`, `save_html()`, the `htmlwidget`, `knitr` and `Shiny`) as the `globals` `window.MAIDR_DOTPAD_SDK_URL` and `window.MAIDR_DOTPAD_ASSET_BASE_URL`, ahead of `maidr.js`, which reads them when a DotPad is connected. Nothing is written when neither is set. Without them a DotPad needs network access to jsDelivr on first connect, even from a `use_cdn = FALSE` document; the rest of the document works offline either way.

The other way to keep a DotPad off the network is to download the SDK once with `maidr_download_dotpad_sdk()`: `show()` and `save_html()` then copy it into `lib/dotpad-sdk-<version>/` beside every `use_cdn = FALSE` document and declare the `globals` with that relative path. A configured URL wins over a downloaded copy. The `widget` and `Shiny` paths render their charts in `srcdoc` frames, where a relative path has nothing to resolve against, and a knitted document does not carry the downloaded copy, so they use only the URL options.

Languages other than English

MAIDR.js reads a chart in English on its own, and in Korean, Japanese, Chinese, Spanish, German, French, Italian or Hindi from a locale pack, `locale-<code>.js`, that it fetches from beside itself when the reader's language is chosen or detected. A document that loads MAIDR.js from the CDN finds the packs there. This package does not bundle them – they would add 0.8 MB to an installed package already over CRAN's size guideline – so every document that loads the bundled MAIDR.js, from `lib/` or `inlined`, declares `window.maidrLocaleBaseUrl` ahead of it: the packs of the bundled version on jsDelivr, unless `maidr.locale_base_url` names another place or none. English needs

no network; another language is fetched when the reader is online and stays English when they are not. A page that declares `window.maidrLocaleBaseUrl` itself keeps its own.

Which MAIDR.js the CDN serves

Documents that load MAIDR.js from the jsDelivr CDN – `show()` and `save_html()` with `use_cdn = TRUE`, and the widget and Shiny paths when they find the machine online – load the latest published MAIDR.js, as the Python binding does. Documents rendered with `use_cdn = FALSE` load the copy bundled with this package, and reach the network only for the two things described above: a DotPad’s SDK and a language other than English. So does a knitted R Markdown or Quarto page, which loads the bundled copy once for all its charts; only the charts kept in frames of their own, in outputs such as an HTML fragment or pagedown, load the CDN copy when the machine is online.

The first CDN document in an R session asks which version is the latest: jsDelivr’s data API (<https://data.jsdelivr.com/v1/packages/npm/maidr/resolved?specifier=latest>), then the npm registry (<https://registry.npmjs.org/-/package/maidr/dist-tags>) if that fails, within `maidr.cdn_timeout` seconds for the two together. The answer is kept for the rest of the session, and the document names that version, which never changes under the reader. When neither answers – offline, blocked, timed out – that too is kept, no error is raised, and the document names the version bundled with this package, the copy `use_cdn = FALSE` would serve, as the Python binding does. So does an answer older than the bundled version. The mutable `maidr@latest` tag, which jsDelivr caches for up to a week, is used only when `maidr.cdn_version = "latest"` asks for it.

Pin a version when a document has to load the same MAIDR.js wherever and whenever it is opened, or to keep the one this package was tested with:

```
# The version bundled with this package, no lookup
options(maidr.cdn_version = "bundled")

# A particular release
options(maidr.cdn_version = "4.9.0")

# Or from the shell, for every session
# MAIDR_CDN_VERSION=bundled Rscript render.R
```

maidr_download_dotpad_sdk

Download the DotPad SDK for use offline

Description

maidr.js drives a **DotPad tactile display** through the vendor’s SDK, which it does not bundle: the braille engine inside it is a 14 MB liblouis build, and every document would carry it for the few readers who own the device. By default maidr.js imports the vendor’s published copy from jsDelivr, pinned to a commit, the first time a DotPad is connected – the one path an offline document (`use_cdn = FALSE`) still takes to the network.

Usage

```
maidr_download_dotpad_sdk(
  dir = maidr_dotpad_sdk_dir(),
  force = FALSE,
  quiet = FALSE
)
```

Arguments

<code>dir</code>	Where to write. Defaults to the option <code>maidr.dotpad_sdk_dir</code> , the environment variable <code>MAIDR_DOTPAD_SDK_DIR</code> , or a per-user cache directory.
<code>force</code>	Refetch files that are already present and correct.
<code>quiet</code>	Say nothing about what was fetched.

Details

This fetches that pinned copy once – the module, the liblouis build, and the LGPL licence text and wrapper sources the vendor asks redistributors to keep beside it – verifying every file against its recorded size and digests (MD5, and SHA-256 on R 4.5 or later), and writes a `manifest.json` beside them naming the commit they came from. From then on [show\(\)](#) and [save_html\(\)](#) copy it into `lib/dotpad-sdk-<version>/` next to every `use_cdn = FALSE` document and tell `maidr.js` where it is, so a reader connects a DotPad without the network. A file already present and correct is left alone, so a second call costs nothing.

A page served from somewhere else – an intranet host, or a knitted document, which does not carry the downloaded copy – names its copy by URL instead, through the options `maidr.dotpad_sdk_url` and `maidr.dotpad_asset_base_url`; see [maidr-options](#). A configured URL wins over a downloaded copy.

Value

The directory, invisibly.

See Also

[maidr-options](#) for naming a copy by URL

Examples

```
## Not run:
maidr_download_dotpad_sdk() # about 14 MB, once
save_html(p, "chart.html", use_cdn = FALSE) # carries the SDK in lib/

## End(Not run)
```

maidr_gantt	<i>Declare that a rectangle layer draws a schedule</i>
-------------	--

Description

`maidr_gantt()` is `ggplot2::geom_rect()` with one thing added: the author saying that these rectangles are intervals in lanes. A declared layer is read as a `gantt` – lanes named, intervals announced, every bar highlightable – where the same rectangles drawn with `geom_rect()` are left unread and cost the whole chart its interactivity.

Nothing about the picture changes. The declaration is carried on the layer object, not in the aesthetics, so the same `xmin/xmax/ymin/ymax` the author would have written to `geom_rect()` produce the same chart: measured on `ggplot2` 3.4.4, the built data is `identical()` to the bare `geom_rect()` layer's and the panel's `x.range` and `y.range` are identical too. Swapping `geom_rect()` for `maidr_gantt()` moves nothing on the page.

The `gantt` layer type [experimental] is one of the experimental plot types: it has not been through a user study, and its reading may change without a deprecation period. See "Experimental Plot Types" in the README.

Usage

```
maidr_gantt(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  lane_axis = c("y", "x"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

<code>mapping</code>	Aesthetics, as for <code>ggplot2::geom_rect()</code> : <code>xmin</code> , <code>xmax</code> , <code>ymin</code> and <code>ymax</code> are required, and every other rectangle aesthetic (<code>fill</code> , <code>colour</code> , <code>alpha</code> , ...) behaves exactly as it does there.
<code>data</code>	The layer's data, as for <code>ggplot2::geom_rect()</code> .
<code>position</code>	Position adjustment, as for <code>ggplot2::geom_rect()</code> .
<code>...</code>	Other arguments passed to the layer, as for <code>ggplot2::geom_rect()</code> – except <code>stat</code> , which that function takes as a formal and this one does not accept. A declared schedule is always drawn from the author's own bounds, so the <code>stat</code> is fixed at <code>"identity"</code> ; measured, a <code>stat</code> written here lands in <code>params</code> , is recognised by neither the <code>geom</code> nor the <code>stat</code> , and is dropped with the warning Ignoring unknown parameters. Aesthetics and <code>geom</code> parameters pass through exactly as they do to <code>ggplot2::geom_rect()</code> – measured, a misspelled aesthetic and a misspelled parameter each raise the identical warning from both.

lane_axis	Which axis the lanes run up: "y" (the default) for the ordinary horizontal schedule – lanes stacked up y, spans running along x – or "x" for the mirror image. It selects which pair of bounds becomes the span and which becomes the lane; it is not a guess the package makes.
na.rm	If FALSE (the default), rows with missing values are removed with a warning.
show.legend	Whether this layer is included in the legends.
inherit.aes	If FALSE, the plot's default aesthetics are not inherited.

Value

A ggplot2 layer, to be added to a plot with `+`.

Why the author is asked

A rectangle layer carries no evidence of what it means. Five structural rules were measured against eight charts on ggplot2 3.4.4, and the best of them – bands partition on the lane axis, more than one band, more than one distinct span, minus a complete-lattice veto – scored 6 of 8 and still claimed a heatmap with one cell missing and a two-region highlight. The table is recorded above the reading itself, in `R/ggplot2_adapter.R`. A monotone waterfall and a one-task-per-lane schedule are the same rectangles, so there is nothing in the geometry to separate; asking the author is the only unfalsified rule.

The consequence is that this is trusted. `maidr_gantt()` over heatmap coordinates announces a heatmap as a schedule, and the package believes it, because any guard strong enough to catch that is the rule the measurements above ruled out.

What it costs not to declare

An undeclared `geom_rect()` layer reads as "unknown", which drops the whole plot to a static image with the "Plot contains unsupported elements" warning. That is unchanged by this function, deliberately: every chart already written keeps exactly the reading it has today.

Lane names

With numeric `ymin/ymax` the lane axis is continuous and has no level names to borrow, so a lane is named by the single explicit tick drawn inside it – `scale_y_continuous(breaks = 1:3, labels = c("design", "build", "test"))` – and by its position on the axis otherwise. A tick whose label is a rendering of its own number is a coordinate rather than a name: measured on the default scale the panel's labels are NA, 1, 2, 3, NA, each one its own break written out, and a lane called "2" says less than a lane called by the position 2 it sits at.

See Also

[save_html\(\)](#) and [show\(\)](#) for rendering the declared chart

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  tasks <- data.frame(
    lane = c(1, 2, 3, 2),
```

```

      start = c(0, 3, 8, 12),
      end = c(3, 8, 11, 15)
    )

    schedule <- ggplot2::ggplot(tasks) +
      maidr_gantt(ggplot2::aes(
        xmin = start, xmax = end,
        ymin = lane - 0.4, ymax = lane + 0.4
      )) +
      ggplot2::scale_y_continuous(
        breaks = 1:3,
        labels = c("design", "build", "test")
      ) +
      ggplot2::labs(x = "week", y = "task")

    # The same rectangles written with `geom_rect()` draw the same chart and
    # are left unread, which costs the plot its interactivity.
    if (interactive()) {
      show(schedule)
    }
  }
}

```

maidr_get_fallback	<i>Get Current MAIDR Fallback Settings</i>
--------------------	--

Description

Retrieves the current fallback configuration for MAIDR.

Usage

```
maidr_get_fallback()
```

Value

A list with the current fallback settings:

- enabled: Logical indicating if fallback is enabled
- format: Character string of the image format
- warning: Logical indicating if warnings are shown

See Also

[maidr_set_fallback\(\)](#) to configure settings

Examples

```
# Get current settings
settings <- maidr_get_fallback()
print(settings)
```

maidr_htmlwidget

Make a plotly, highcharter or echarts4r htmlwidget accessible

Description

Attaches MAIDR to an interactive chart drawn by another R package, so a screen reader user can explore it with the keyboard, hear it as sonification, and read it as text and braille. The chart is read by the MAIDR JavaScript adapter for the library that draws it, once it has been drawn in the browser; nothing about it is recomputed in R.

Usage

```
maidr_htmlwidget(
  widget,
  use_cdn = FALSE,
  percentile_bands = NULL,
  hover_mode = NULL
)
```

Arguments

widget	An htmlwidget created by plotly, highcharter or echarts4r.
use_cdn	Logical. Where the MAIDR scripts are loaded from: - FALSE (default): the copy bundled with this package, which works offline and makes no network request. - TRUE: the jsDelivr CDN, which loads the latest published MAIDR unless <code>maidr.cdn_version</code> pins one; see <code>?"maidr-options"</code>.
percentile_bands	echarts4r only. The fan charts the chart draws, each read as one percentile band layer; see "Fan charts in echarts4r" below. A list of fans, or a single fan, each a list with: - median: the name of the median's line series. - bands: a data frame with columns <code>series</code>, <code>lower</code> and <code>upper</code>, or a list of lists with those elements, one per band: the name of the band's filled series and the quantiles of its lower and upper edges, as fractions. - title, name (optional): the layer's title and name, in place of the ones read from the chart. NULL (default) declares none.

`hover_mode` highcharter and echarts4r only. How the pointer moves the reader through the chart: "pointermove", "click" or "off", as in `show()`, written into the chart MAIDR reads once it has been drawn. NULL (default) takes `getOption("maidr.hover_mode")`, and when that is unset writes nothing, so `maidr.js` uses the reader's own setting. A plotly chart is read by the MAIDR core itself, which takes no hover mode from R: an explicit `hover_mode` is refused for one, and the option is not applied to it.

Details

Supported widgets:

- **plotly**: `plotly::plot_ly()` and `plotly::ggplotly()`. MAIDR's core detects a Plotly chart on the page by itself, so no adapter is added.
- **highcharter**: `highcharter::highchart()`, `highcharter::hchart()` and the stock, map and gantt variants.
- **echarts4r**: `echarts4r::e_charts()`. The chart is switched to ECharts' SVG renderer, since MAIDR highlights the mark being read by finding it among the drawn SVG elements, and the default canvas renderer draws none.

Which chart types each library supports is decided by its MAIDR adapter; see <https://maidr.ai/> for the lists. A chart the adapter cannot read is left as it was drawn, with a warning in the browser console.

The widget keeps working everywhere an `htmlwidget` does: the RStudio viewer, `htmlwidgets::saveWidget()`, R Markdown and Quarto documents, and Shiny (`plotly::renderPlotly()`, `highcharter::renderHighchart()`, `echarts4r::renderEcharts4r()`), where the chart is read again each time the server re-renders it.

Applying it twice returns the widget unchanged.

Value

The widget, with MAIDR attached. Print it, return it from a Shiny render function, or save it as you would the original.

Fan charts in echarts4r

ECharts has no series for a band: a fan chart is drawn as a median `e_line()` and, for each band, an invisible line holding its lower edge with an `e_area()` of its width stacked on it. Nothing in the chart says which quantiles those edges are, so they are stated with `percentile_bands`, and MAIDR then reads the median and its bands as one percentile band layer (experimental): each position announced with its median and the edges of every band, as ECharts computed them for drawing. Name the median's series and each band's filled series by the name (or id) ECharts knows it by – in `echarts4r`, the name given to `e_line()` and `e_area()`, or the column name when none was given. Each band's lower and upper are fractions, `0.05` rather than `5`, every lower below `0.5` and every upper above it, and the bands must nest. Draw the fan over a category x axis, a character or factor column given to `e_charts()`: over a numeric x axis ECharts stacks each series on the x values rather than the y, so neither the picture nor the reading is a band. A width series stacks on a negative lower edge only under `stackStrategy = "all"`.

The option is read by maidr.js 4.15.0 and later, the version bundled with this package. With `use_cdn = TRUE` and a `maidr.cdn_version` pinned below 4.15.0, `percentile_bands` is checked and then ignored with a warning, and the series read as they would be without it.

Examples

```
if (requireNamespace("plotly", quietly = TRUE)) {
  w <- plotly::plot_ly(
    x = c("Mon", "Tue", "Wed"), y = c(20, 14, 23), type = "bar"
  )
  w <- maidr_htmlwidget(w)
}
## Not run:
# Pipe-friendly
library(echarts4r)
mtcars |>
  e_charts(wt) |>
  e_scatter(mpg) |>
  maidr_htmlwidget()

# A fan chart: a median and a 90% band drawn as a stacked area
fan <- data.frame(
  week = paste("Week", 1:8),
  median = c(10, 12, 13, 15, 16, 18, 19, 21),
  lower = c(8, 9, 9, 10, 10, 11, 11, 12)
)
fan$width <- 2 * (fan$median - fan$lower)
fan |>
  e_charts(week) |>
  e_line(median, name = "Median") |>
  e_line(lower, stack = "band", name = "lower", symbol = "none",
    lineStyle = list(opacity = 0)) |>
  e_area(width, stack = "band", name = "90% interval", symbol = "none") |>
  maidr_htmlwidget(
    percentile_bands = list(
      median = "Median",
      bands = data.frame(series = "90% interval", lower = 0.05, upper = 0.95)
    )
  )
## End(Not run)
```

maidr_manhattan

Declare that a point layer draws a Manhattan plot

Description

`maidr_manhattan()` is `ggplot2::geom_point()` with three things added: the author saying that the points are a Manhattan plot, the significance threshold the chart is read against, and two aesthetics for what each point is – `label` (the SNP or variant) and `region` (its chromosome). A declared

layer is read as a manhattan: the description says how many points clear the threshold and names them, a rotor mode steps through only those points, and every point is announced with its label and chromosome. The same points drawn with `geom_point()` read as a scatter of tens of thousands of positions, walked one at a time.

Nothing about the picture changes: the geom draws exactly what `geom_point()` draws, and neither label nor region reaches a mark.

The manhattan layer type [experimental] is one of the experimental plot types: it has not been through a user study, and its reading may change without a deprecation period. See "Experimental Plot Types" in the README.

Usage

```
maidr_manhattan(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  significance,
  direction = c("above", "below"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Aesthetics, as for <code>ggplot2::geom_point()</code> : <code>x</code> (the position along the genome) and <code>y</code> (the significance, such as $-\log_{10}$ of the <code>p</code> value) are required. <code>label</code> may name each point (the SNP) and <code>region</code> its chromosome; both are announced with the point. Every other point aesthetic (colour, size, alpha, ...) behaves exactly as it does there.
data	The layer's data, as for <code>ggplot2::geom_point()</code> .
position	Position adjustment, as for <code>ggplot2::geom_point()</code> .
...	Other arguments passed to the layer, as for <code>ggplot2::geom_point()</code> – except <code>stat</code> , which is fixed at "identity": a declared layer is always drawn from the author's own points.
significance	The significance cutoff on the <code>y</code> axis, one number in the units <code>y</code> is announced in. Required.
direction	Which side of significance the significant points are on: "above" (the default) for a $-\log_{10}(p)$ axis, "below" for a raw <code>p</code> value.
na.rm	If FALSE (the default), rows with missing values are removed with a warning.
show.legend	Whether this layer is included in the legends.
inherit.aes	If FALSE, the plot's default aesthetics are not inherited.

Details

x is the position along the genome – usually each variant’s position offset by the length of the chromosomes before it, with the axis labelled by chromosome through `scale_x_continuous(breaks = , labels = )` – and y the significance. A point is significant when its y is at or above significance (at or below it with `direction = "below"`), in the units y is announced in: `significance = -log10(5e-8)` for genome-wide significance on a $-\log_{10}(p)$ axis. There is no default, for the reason `maidr_volcano()` gives.

Value

A `ggplot2` layer, to be added to a plot with `+`.

See Also

`maidr_volcano()` for a volcano plot; `save_html()` and `show()` for rendering the declared chart

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  snps <- data.frame(
    snp = paste0("rs", 1:8),
    chr = rep(c("1", "2"), each = 4),
    pos = c(1, 2, 3, 4, 6, 7, 8, 9),
    p = c(0.3, 2e-9, 0.01, 0.6, 0.05, 0.8, 4e-10, 0.2)
  )

  manhattan <- ggplot2::ggplot(
    snps,
    ggplot2::aes(x = pos, y = -log10(p), colour = chr)
  ) +
    maidr_manhattan(
      ggplot2::aes(label = snp, region = chr),
      significance = -log10(5e-8)
    ) +
    ggplot2::geom_hline(yintercept = -log10(5e-8), linetype = "dashed") +
    ggplot2::scale_x_continuous(breaks = c(2.5, 7.5), labels = c("1", "2")) +
    ggplot2::labs(x = "Chromosome", y = "-log10 p")

  if (interactive()) {
    show(manhattan)
  }
}
```

Description

Disables automatic MAIDR rendering and restores normal plot behavior. After calling this, Base R plots display in the standard graphics window, ggplot2 objects render with the default ggplot2 method, and lattice charts print as lattice draws them: `lattice.options(print.function = )` is set back to what it was before `maidr` set it. In an R Markdown or Quarto document, the chunks after it are knitted as they would be without `maidr`, and `maidr`'s knitr hooks are taken out until `maidr_on()`.

Usage

```
maidr_off()
```

Details

It lasts for the R session, not only the document that calls it: the documents rendered after it in the same session – the later vignettes of R CMD build, which renders them all in one process – and the console stay off too. A document that turns `maidr` off should end with `maidr_on()`.

Value

Invisible TRUE on success

See Also

`maidr_on()` to enable MAIDR rendering

maidr_on

Enable MAIDR Plot Interception

Description

Turns on the accessible rendering of ggplot2, lattice and Base R plots. It is on after `library(maidr)`: `maidr_on()` is needed only to turn it back on after `maidr_off()`.

Usage

```
maidr_on()
```

Details

Interception is on by default after `library(maidr)`: printing a ggplot2 or lattice object opens it in the MAIDR viewer, and Base R plotting calls are recorded until `show()` is called. lattice is reached through its own hook, `lattice.options(print.function = )`, which `maidr` sets once lattice's namespace is loaded; a print function set before is kept, and draws the prints `maidr` leaves to lattice.

In an R Markdown or Quarto document, `library(maidr)` is enough: loading `maidr` during the knit, or else the first chart the document draws, installs the knitr hooks that make every plot of it an accessible chart, in each render of a session. So any document rendered in a session where `maidr` is

loaded, even only its namespace (as a package that imports maidr loads it), gets charts, whether or not it loads maidr itself; `options(maidr.auto_show = FALSE)` before the render prevents that. In a document, `maidr_on()` installs the hooks at once.

In HTML output the charts are part of the page, which loads `maidr.js` once for all of them, and the static figures of a chunk are recorded with `svglite` rather than knitr's default `png` (`maidr.knitr_dev` in `maidr-options` turns that off). A chunk can draw several charts – `print(p)` in a loop, several Base R charts – and each takes the place of its own figure; a figure maidr cannot read as one chart stays knitr's image. HTML that cannot hold a chart in the page, such as an HTML fragment or pagedown, shows each chart in a frame of its own. In PDF, Word, Markdown, EPUB or xaringan output the plots are knitr's figures, as without maidr. A chart a document returns or prints never opens the viewer; an explicit `show()` still does, during the render, and puts nothing in the page.

A chart is drawn at its chunk's `fig.width` and `fig.height`, in inches, as knitr draws the chunk's figures: `fig.asp` and `fig.dim` set them as they do for a figure, and so do R Markdown's `fig_width` and `fig_height` and Quarto's `fig-width` and `fig-height`. `html_document` and Quarto's HTML formats draw figures at 7 x 5 in, maidr's own size outside a document. A format with a figure size of its own draws its charts at that size: `rmarkdown::html_vignette` at 3 x 3 in, small enough that a Base R chart shows fewer tick labels, `ioslides` at 7.5 x 4.5 in, `slidy` at 8 x 6 in and `knitr::knit()` on its own at 7 x 7 in. Set `fig.width` and `fig.height` to draw a chart at another size.

A chart is shown at the size knitr shows the chunk's figure: `fig.width` times `dpi` over `fig.retina` pixels wide for a retina figure, 96 pixels to the inch in R Markdown's and Quarto's HTML, where knitr's figures are `png` at `dpi = 96` and `fig.retina = 2`; a pixel to a dot for any other bitmap; and at the size its file gives an `svg` figure, 96 pixels to the inch for `svglite`'s and 72 for R's `svg()` with Cairo 1.18 (Quarto's `fig-format: svg`). So `dpi`, Quarto's `fig-dpi`, changes it as it changes a figure. Its SVG's `viewBox` is still 72 to the inch, the canvas it was drawn on, and what maidr reads out is the same at any size. `show()` and `save_html()` show a chart at 72 pixels to the inch.

A candlestick chart is drawn at least 12 x 6 in. A chunk that sets a smaller size of its own is told so in a message (see `show()`): among the chunk's messages for a `ggplot2` chart, and on the console for a Base R `chartSeries()` chart, which maidr reads once the chunk has run. The document's figure size, which the format, its YAML or `knitr::opts_chunk$set()` sets for every chunk, was not asked of a candlestick chart, which is drawn larger than it without a message.

maidr draws a Base R chart again from its recorded calls, with the margins and text size its chunk's `par()` calls set (`mar`, `mai`, `oma`, `omi`, `mex` and `cex`), so it fits where R drew it. A chart R drew in its chunk with settings maidr does not record – a `par()` called by name, as `graphics::par()`, or another of its settings – can still be too small for its size in maidr. Too small for the document's figure size, it is drawn larger, as one too small for 7 x 5 in is outside a document (see `show()`), and a message on the console names the size. Too small for a size its chunk sets of its own, it stays knitr's picture, with a warning naming the size.

Whether a chunk set its size is read from the size, not from where it was set: a chunk that sets the document's own, `fig.width = 7` and `fig.height = 5` in `html_document`, is taken to set none, for a candlestick chart and a Base R chart alike.

The size is the room the chart is laid out in, not what a reader hears: its data, titles and axes are the same at every size, though a lattice chart conditioned on one variable with no `layout =` arranges its panels, and the grid a reader moves through, for it (see `show()`).

`out.width` sets the width a chart is shown at, wider or narrower than its own size, as it does a figure's, and `out.height` its height, though a chart keeps its shape where an image is stretched; `fig.align` aligns it. In Quarto a captioned chart is centred unless the chunk aligns it, a percentage

out-width in a layout panel is its cell's, and column: screen and its kin stretch a chart across the screen. A chart always shrinks to fit a page narrower than itself, keeping its shape: Quarto's `fig-responsive: false`, which keeps an image from shrinking, is not read, so that a reader on a phone or with the page magnified need not scroll sideways through a chart.

In a Quarto `reveal.js` deck a chart alone on a slide is stretched to the room left on the slide, as Quarto stretches a lone figure, less the room `maidr`'s instructions, or a line of its text and its braille field, take below it, so that they stay on the slide; it is a little smaller than the figure for that. `auto-stretch` is read from the document's own front matter, not from a project's `_quarto.yml`; a slide marked `.nostretch`, a chart given an out-height, one in columns, a panel or a fragment, and a phone's scroll view are left alone, as Quarto leaves an image. An `ioslides` chart too large to leave that room on its slide is made smaller. In a Quarto dashboard a chart fills its card as an image does, less a line for `maidr`'s text below it; the card scrolls to `maidr`'s braille and review fields. The size a chart is given on a slide or in a card does not change while a reader uses it.

A chart's caption, in R Markdown and bookdown, is plain text: Markdown, maths and `\@ref()` in `fig.cap` are shown as written, and a bookdown text reference (`fig.cap = "(ref:label)"`) brings them in. `fig.show = "animate"` stays knitr's animation.

A chunk cached with `cache = TRUE` brings its charts back from knitr's cache. One cached with `cache = 1` or `cache = 2` shows them only when its code runs: rendered again from the cache, its figures are knitr's static images, since the charts they were drawn from are not cached with them.

Value

Invisible TRUE on success

See Also

[maidr_off\(\)](#) to disable MAIDR rendering

Examples

```
library(maidr)

# Enable interception (on by default after library(maidr))
maidr_on()

# Now all plots render as accessible MAIDR charts
library(ggplot2)
ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar()

barplot(table(mtcars$cyl))
```

maidr_output	<i>MAIDR Output Container for Shiny UI</i>
--------------	--

Description

Creates a Shiny output container for MAIDR widgets using htmlwidgets. This provides automatic dependency injection and robust JavaScript initialization.

Usage

```
maidr_output(output_id, width = "100%", height = "400px")
```

Arguments

output_id	The output variable to read the plot from
width	The width of the plot container (default: "100percent")
height	The height of the plot container (default: "400px")

Value

A Shiny widget output function for use in UI

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(maidr_output("myplot"))
}
```

maidr_pr_curve	<i>Declare that a path layer draws a precision-recall curve</i>
----------------	---

Description

`maidr_pr_curve()` is `ggplot2::geom_path()` with three things added: the author saying that the path is a precision-recall curve, a threshold aesthetic for the decision threshold each point was scored at, and the share of positives in the data – the precision a classifier that guesses keeps at every recall, which every point of the curve is read against. A declared layer is read as a `pr_curve`: each point announced as its recall and precision, its threshold and how far its precision sits above that baseline, with the average precision of each curve and the point with the best F1 score in the description. The same path drawn with `geom_path()` or `geom_line()` reads as a line, which says the rates and nothing a precision-recall curve is drawn to say.

Nothing about the picture changes: the `geom` draws exactly what `geom_path()` draws, and `threshold` reaches no mark.

The `pr_curve` layer type [experimental] is one of the experimental plot types: it has not been through a user study, and its reading may change without a deprecation period. See "Experimental Plot Types" in the README.

Usage

```
maidr_pr_curve(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  prevalence = NULL,
  ap = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Aesthetics, as for <code>ggplot2::geom_path()</code> : x (the recall) and y (the precision) are required, and threshold may name the decision threshold at each point. Every other path aesthetic (colour, linetype, group, ...) behaves exactly as it does there.
data	The layer's data, as for <code>ggplot2::geom_path()</code> .
position	Position adjustment, as for <code>ggplot2::geom_path()</code> .
...	Other arguments passed to the layer, as for <code>ggplot2::geom_path()</code> – except stat, which is fixed at "identity".
prevalence	The share of positives in the data each curve was scored on, from 0 to 1: the height of the chance baseline. One number for a single curve or for curves scored on the same data; for several scored on different data, a vector named by the groups' names, or unnamed and in the groups' sorted order. NULL (the default) reads the curve without a baseline.
ap	The average precision of each curve as the author computed it – with <code>yardstick::average_precision()</code> say – announced in the description in place of the step-wise area over the drawn points. Given as prevalence is. NULL (the default) measures it from the points.
na.rm	If FALSE (the default), rows with missing values are removed with a warning.
show.legend	Whether this layer is included in the legends.
inherit.aes	If FALSE, the plot's default aesthetics are not inherited.

Value

A `ggplot2` layer, to be added to a plot with `+`.

What is asked of the data

x is the recall and y the precision, both fractions of one, in any order: the average precision is measured over the points sorted by recall. Several classifiers on one chart are several groups, split by colour, linetype or group as a multi-series line is, and each is announced by its group's name.

Curves maidr reads without a declaration

`ggplot2::autoplot()` of a `yardstick::pr_curve()` draws a `geom_path()` that maps recall against precision, and is read as a precision-recall curve as it stands – as is any `geom_line()` or `geom_path()` of columns named recall and precision. Such a curve carries neither the thresholds nor the share of positives into the plot, so no threshold is announced, the average precision is measured from the points, and the points are not read against a baseline; `maidr_pr_curve()` is how an author supplies them.

A Base R `plot(recall, precision, type = "l")` (or `type = "s"`) and a lattice `xyplot(precision ~ recall, type = "l")` are read the same way when their axes are titled Recall and Precision – which both take from the variables' names unless `xlab` and `ylab` say otherwise – and every value is a fraction of one.

Until the bundled `maidr.js` carries the trace

The `pr_curve` trace shipped in `maidr.js` 4.14.0. While the copy this package bundles is older (see `maidr::MAIDR_VERSION`), a declared or detected curve is read as a line, so that every chart keeps rendering.

See Also

`maidr_roc()`, the ROC curve's declaration; `save_html()` and `show()` for rendering the declared chart

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  curve <- data.frame(
    recall = c(0, 0.2, 0.4, 0.6, 0.8, 1),
    precision = c(1, 1, 0.89, 0.8, 0.62, 0.3),
    cutoff = c(1, 0.9, 0.75, 0.6, 0.4, 0)
  )

  pr <- ggplot2::ggplot(curve) +
    maidr_pr_curve(
      ggplot2::aes(x = recall, y = precision, threshold = cutoff),
      prevalence = 0.3
    ) +
    ggplot2::geom_hline(yintercept = 0.3, linetype = "dashed") +
    ggplot2::labs(x = "Recall", y = "Precision")

  if (interactive()) {
    show(pr)
  }
}
```

maidr_roc

Declare that a path layer draws a ROC curve

Description

`maidr_roc()` is `ggplot2::geom_path()` with two things added: the author saying that the path is a receiver operating characteristic curve, and a `threshold` aesthetic for the decision threshold each point was scored at. A declared layer is read as a roc – each point announced as its false and true positive rates, its threshold and its height above the chance diagonal, the area under each curve and the best operating point in the description – where the same path drawn with `geom_path()` or `geom_line()` reads as a line, which says the rates and nothing a ROC curve is drawn to say.

Nothing about the picture changes: the geom draws exactly what `geom_path()` draws, and `threshold` reaches no mark.

The roc layer type [experimental] is one of the experimental plot types: it has not been through a user study, and its reading may change without a deprecation period. See "Experimental Plot Types" in the README.

Usage

```
maidr_roc(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  auc = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

<code>mapping</code>	Aesthetics, as for <code>ggplot2::geom_path()</code> : <code>x</code> (the false positive rate) and <code>y</code> (the true positive rate) are required, and <code>threshold</code> may name the decision threshold at each point. Every other path aesthetic (<code>colour</code> , <code>linetype</code> , <code>group</code> , ...) behaves exactly as it does there.
<code>data</code>	The layer's data, as for <code>ggplot2::geom_path()</code> .
<code>position</code>	Position adjustment, as for <code>ggplot2::geom_path()</code> .
<code>...</code>	Other arguments passed to the layer, as for <code>ggplot2::geom_path()</code> – except <code>stat</code> , which is fixed at "identity": a declared curve is always drawn from the author's own rates.
<code>auc</code>	The area under each curve as the author computed it – with <code>pROC::auc()</code> , <code>yardstick::roc_auc()</code> or by hand – announced in the description in place of the trapezoid rule over the drawn points. One number for a single curve; for several, a vector named by the groups' names, or unnamed and in the groups'

	sorted order. NULL (the default) measures the area from the points, which is what the trapezoid rule gives and what those functions compute for an empirical curve.
<code>na.rm</code>	If FALSE (the default), rows with missing values are removed with a warning.
<code>show.legend</code>	Whether this layer is included in the legends.
<code>inherit.aes</code>	If FALSE, the plot's default aesthetics are not inherited.

Value

A `ggplot2` layer, to be added to a plot with `+`.

What is asked of the data

`x` is the false positive rate and `y` the true positive rate, both fractions of one, in the order the curve is to be walked – from (0, 0) up, as `sklearn.metrics.roc_curve()` returns them, or from (1, 1) down, as `pROC::coords()` does; the area is measured over the points sorted by `x` either way. Several classifiers on one chart are several groups, split by colour, linetype or group as a multi-series line is, and each is announced by its group's name.

Curves `maidr` reads without a declaration

Two idioms name their axes after the ROC's own vocabulary, and are read as ROC curves as they stand: `pROC::ggroc()`, whose `geom_line()` maps specificity (or 1-specificity with `legacy.axes = TRUE`) against sensitivity, and `ggplot2::autoplot()` of a `yardstick::roc_curve()`, whose `geom_path()` maps 1 – specificity against sensitivity. Where `x` is specificity itself – `pROC`'s default, drawn on a reversed axis – the announced rate is 1 – specificity and the axis is named so, because the height above chance is measured against the false positive rate and a rate read off a reversed axis would put every point on the wrong side of the diagonal. Neither idiom carries thresholds or the area into the plot, so the area is measured from the points and no threshold is announced; `maidr_roc()` is how an author supplies both.

Until the bundled `maidr.js` carries the trace

The roc trace shipped in `maidr.js` 4.9.0. While the copy this package bundles is older (see `maidr::MAIDR_VERSION`), a declared or detected curve is read as a line, so that every chart keeps rendering; the reading switches to roc with the next bundle update and no change to the chart.

What it costs not to declare

A `geom_line()` of rates under any other column names keeps the line reading it has today, deliberately: every chart already written keeps exactly the reading it has.

See Also

`maidr_gantt()`, the other per-layer declaration; `save_html()` and `show()` for rendering the declared chart

Examples

```

if (requireNamespace("ggplot2", quietly = TRUE)) {
  curve <- data.frame(
    fpr = c(0, 0.05, 0.1, 0.2, 0.35, 0.6, 1),
    tpr = c(0, 0.55, 0.75, 0.86, 0.93, 0.98, 1),
    cutoff = c(1, 0.8, 0.6, 0.45, 0.3, 0.15, 0)
  )

  roc <- ggplot2::ggplot(curve) +
    maidr_roc(ggplot2::aes(x = fpr, y = tpr, threshold = cutoff)) +
    ggplot2::geom_abline(linetype = "dashed") +
    ggplot2::labs(x = "False positive rate", y = "True positive rate")

  # The same path written with `geom_path()` draws the same chart and is
  # read as a line: the rates, and none of what a ROC curve is read for.
  if (interactive()) {
    show(roc)
  }
}

```

maidr_set_fallback	<i>Configure MAIDR Fallback Behavior</i>
--------------------	--

Description

Configure how MAIDR handles unsupported plot types or layers. When fallback is enabled, unsupported plots are rendered as static images instead of failing or returning empty data.

Usage

```
maidr_set_fallback(enabled = NULL, format = NULL, warning = NULL)
```

Arguments

enabled	Logical. If TRUE, unsupported plots fall back to image rendering. If FALSE, unsupported layers return empty data. If NULL (default), the current setting is kept.
format	Character. Image format for fallback: "png", "svg", or "jpeg". If NULL (default), the current setting is kept.
warning	Logical. If TRUE, shows a warning message when falling back to image rendering. If NULL (default), the current setting is kept.

Value

Invisibly returns a list of the previous settings.

See Also

`maidr_get_fallback()` to retrieve current settings

Examples

```
# Save current settings and restore on exit
old_settings <- maidr_get_fallback()
on.exit(maidr_set_fallback(
  enabled = old_settings$enabled,
  format = old_settings$format,
  warning = old_settings$warning
))

# Disable fallback (unsupported plots will have empty data)
maidr_set_fallback(enabled = FALSE)

# Use SVG format for fallback images
maidr_set_fallback(format = "svg")

# Disable warning messages
maidr_set_fallback(warning = FALSE)

# Configure multiple options
maidr_set_fallback(enabled = TRUE, format = "png", warning = TRUE)
```

maidr_survival	<i>Declare a Kaplan-Meier curve</i>
----------------	-------------------------------------

Description

`maidr_survival()` draws Kaplan-Meier survival curves – one per stratum, each held level until an event drops it – and declares them as such, so that `maidr` reads them as a `survival` layer: each time announced with its survival estimate and, when the band is drawn, its confidence interval, the times a subject was censored marked so a reader can step between them with the rotor, and the median survival of each curve in the description. The same staircase drawn with `geom_step()` reads as a step chart, which announces the levels and nothing of the censoring that says how much of the curve the data still supports.

Give it a `survival::survfit` object and it draws the curves `plot()` of that object would: `ggplot() + maidr_survival(fit)`. Give it data and aesthetics and it draws them as they are, as any `geom` does.

The survival layer type [experimental] is one of the experimental plot types: it has not been through a user study, and its reading may change without a deprecation period. See "Experimental Plot Types" in the README.

Usage

```
maidr_survival(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  conf.int = FALSE,
  conf.int.alpha = 0.2,
  censor = TRUE,
  censor.shape = 3,
  censor.size = 3,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Aesthetics, as for <code>ggplot2::geom_step()</code> : x (the time) and y (the survival estimate) are required, and ymin, ymax and censored may be mapped as described in Details. Or a survfit object, drawn as described there.
data	The layer's data, as for <code>ggplot2::geom_step()</code> , or a survfit object. When mapping is the survfit object, data may be further aesthetics.
position	Position adjustment, as for <code>ggplot2::geom_step()</code> .
...	Other arguments passed to the layer, such as linewidth or a fixed colour – except stat, which is fixed at "identity".
conf.int	Whether to draw the confidence band from ymin and ymax (from the object's own bounds, for a survfit), and read it as each point's yMin and yMax. FALSE by default.
conf.int.alpha	The band's opacity.
censor	Whether to draw a mark at each censored time. Censoring is read either way.
censor.shape, censor.size	The marks' plotting character and size, as shape and size are given to <code>ggplot2::geom_point()</code> .
na.rm	If FALSE (the default), rows with missing values are removed with a warning.
show.legend	Whether this layer is included in the legends.
inherit.aes	If FALSE, the plot's default aesthetics are not inherited. Always FALSE for a survfit object.

Value

A ggplot2 layer, to be added to a plot with `+`.

From a survfit object

The curves start at the origin `survival::survfit0()` adds, as `plot()` starts them, and run to each stratum's last time. The layer draws from a data frame of its own, with the columns Time,

Survival probability, lower, upper, censored and, for a fit with strata, Stratum, which are what the axes and the legend are titled by; map further aesthetics onto them, `maidr_survival(fit, aes(linetype = Stratum))`, and retile with `ggplot2::labs()`. The plot's own aesthetics are not inherited. A multi-state fit, or one with several curves per stratum (`survfit()` of a Cox model over several new subjects), is an error: neither is one survival curve per stratum.

From data

`x` is the time and `y` the survival estimate, one row per time at which the estimate is reported, and the strata are groups, split by colour, `linetype` or group as the series of a line chart are. `censored` marks the times a subject left the study without the event: `TRUE`, or a count greater than zero such as a `survfit` summary's `n.censor`. `ymin` and `ymax` are the band's bounds at each time, drawn and read only with `conf.int = TRUE`.

What is read

One series per stratum, each point a row: the time, the estimate, censored where the row is censored, and, when the band is drawn, `yMin` and `yMax`. Censoring is read whether or not the marks are drawn, as it is from a Base R plot: it is part of the estimate, not of its decoration.

survminer

A figure from `survminer::ggsurvplot()` is read the same way when it is handed to `show()` or `save_html()` as it stands: its curve, band and censoring marks are redrawn as one `maidr_survival()` layer from the data the figure carries. The risk table, and a figure whose curve is not a survival probability (`fun = "event" or "cumhaz"`), are not read; the first is left out, the second keeps the reading `geom_step()` has.

See Also

`maidr_roc()` and `maidr_volcano()`, the other declarations; `save_html()` and `show()` for rendering the declared chart

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("survival", quietly = TRUE)) {
  fit <- survival::survfit(
    survival::Surv(time, status) ~ sex,
    data = survival::lung
  )

  km <- ggplot2::ggplot() +
    maidr_survival(fit, conf.int = TRUE) +
    ggplot2::labs(x = "Days", title = "Survival by sex")

  # The same curves drawn with `geom_step()` read as a step chart: the
  # levels, and nothing of who was censored.
  if (interactive()) {
    show(km)
  }
}
```

```
}
```

```
maidr_volcano
```

Declare that a point layer draws a volcano plot

Description

`maidr_volcano()` is `ggplot2::geom_point()` with three things added: the author saying that the points are a volcano plot, the significance threshold the chart is read against, and two aesthetics for what each point is – `label` (the gene) and `region` (any grouping announced alongside it, such as up- or down-regulated). A declared layer is read as a volcano: the description says how many points clear the threshold and names them, a rotor mode steps through only those points, and every point is announced with its label and region. The same points drawn with `geom_point()` read as a scatter, which says two coordinates and withholds which gene they belong to.

Nothing about the picture changes: the geom draws exactly what `geom_point()` draws, and neither `label` nor `region` reaches a mark. Draw the threshold lines yourself, with `geom_hline()` and `geom_vline()`; `maidr` leaves reference lines out of the reading.

The volcano layer type [experimental] is one of the experimental plot types: it has not been through a user study, and its reading may change without a deprecation period. See "Experimental Plot Types" in the README.

Usage

```
maidr_volcano(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  significance,
  effect = NULL,
  direction = c("above", "below"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

<code>mapping</code>	Aesthetics, as for <code>ggplot2::geom_point()</code> : <code>x</code> (the effect size, such as a log2 fold change) and <code>y</code> (the significance, such as $-\log_{10}$ of the p value) are required. <code>label</code> may name each point (the gene) and <code>region</code> the group it belongs to; both are announced with the point. Every other point aesthetic (<code>colour</code> , <code>size</code> , <code>alpha</code> , ...) behaves exactly as it does there.
<code>data</code>	The layer's data, as for <code>ggplot2::geom_point()</code> .
<code>position</code>	Position adjustment, as for <code>ggplot2::geom_point()</code> .

...	Other arguments passed to the layer, as for <code>ggplot2::geom_point()</code> – except <code>stat</code> , which is fixed at "identity": a declared layer is always drawn from the author's own points.
significance	The significance cutoff on the y axis, one number in the units y is announced in. Required.
effect	The effect-size cutoff, applied to the magnitude of x, or NULL (the default) for none.
direction	Which side of significance the significant points are on: "above" (the default) for a $-\log_{10}(p)$ axis, "below" for a raw p value.
na.rm	If FALSE (the default), rows with missing values are removed with a warning.
show.legend	Whether this layer is included in the legends.
inherit.aes	If FALSE, the plot's default aesthetics are not inherited.

Value

A `ggplot2` layer, to be added to a plot with `+`.

The threshold is stated, never guessed

A point is significant when its y is at or above `significance` (at or below it with `direction = "below"`) and, when `effect` is given, the magnitude of its x is at least `effect` – a fold change of -2 is as large an effect as one of +2. Both are in the units the axes are announced in, which are the data's own: `significance =  $-\log_{10}(0.05)$` for `y =  $-\log_{10}(\text{pvalue})$` , and `significance = 0.05`, `direction = "below"` for a raw p value, even on a log-scaled y axis, whose points are announced as the p values they are.

There is no default, because these charts sit on transformed axes whose conventions differ by field and by tool, and a line drawn in the wrong place would put every point on the wrong side of it without saying so.

See Also

`maidr_manhattan()` for a Manhattan plot; `save_html()` and `show()` for rendering the declared chart

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  genes <- data.frame(
    gene = c("TP53", "MYC", "EGFR", "BRCA1", "KRAS", "PTEN", "GAPDH"),
    log2fc = c(2.8, -3.1, 1.6, -0.4, 0.3, -2.2, 0.1),
    padj = c(1e-8, 1e-6, 0.02, 0.4, 0.7, 1e-4, 0.9)
  )

  volcano <- ggplot2::ggplot(
    genes,
    ggplot2::aes(x = log2fc, y = -log10(padj))
  ) +
    maidr_volcano(
      ggplot2::aes(label = gene),
```

```

      significance = -log10(0.05),
      effect = 1
    ) +
    ggplot2::geom_hline(yintercept = -log10(0.05), linetype = "dashed") +
    ggplot2::geom_vline(xintercept = c(-1, 1), linetype = "dashed") +
    ggplot2::labs(x = "log2 fold change", y = "-log10 adjusted p")

    if (interactive()) {
      show(volcano)
    }
  }
}

```

map_visual_to_dom_panel

Map visual panel position to DOM panel name

Description

This function handles the mismatch between visual layout order (row-major) and DOM element generation order (column-major) in gridSVG.

Usage

```
map_visual_to_dom_panel(panel_info, gtable)
```

Arguments

panel_info	Panel information from layout
gtable	Gtable object

Details

Visual layout (row-major): 1 2 3 4

DOM order (column-major): 1 3 2 4

Value

Gtable panel name or NULL if not found

organize_facet_grid	<i>Organize subplots into 2D grid structure</i>
---------------------	---

Description

Organize subplots into 2D grid structure

Usage

```
organize_facet_grid(subplots, panel_layout)
```

Arguments

subplots	List of processed subplot data
panel_layout	Panel layout information

Value

2D grid structure

pheatmap	<i>Draw a clustered heat map with pheatmap, keeping its values for maidr</i>
----------	--

Description

A wrapper of `pheatmap::pheatmap()` that `library(maidr)` puts on the search path. It calls the original with every argument unchanged, draws exactly what it draws and returns what it returns, with one addition: the matrix the chart was drawn from, kept with the result so that `show()` and `save_html()` announce each cell's exact value, and its annotation data frames, so that every annotation track is read exactly too.

Usage

```
pheatmap(mat, ...)
```

Arguments

mat	The numeric matrix to draw, as <code>pheatmap::pheatmap()</code> takes it
...	Every other argument of <code>pheatmap::pheatmap()</code> , passed on unchanged

Details

`pheatmap::pheatmap()` returns its trees and its drawing but not its matrix. Without this wrapper – a chart drawn by a call to `pheatmap::pheatmap()` by name, before `maidr` was loaded, or with `'pheatmap'` attached after `maidr`, which then masks this function – `maidr` can only read each cell back from its colour, to within one band of the legend, and says so.

The matrix kept is the one drawn: the input after `scale = "row" or "column"`, computed as `pheatmap` computes it; the k-means centres with `kmeans_k`; in the order of the row and column trees. The annotations kept are `annotation_row`, `annotation_col` (or the older `annotation`) and `annotation_colors` as passed; they are matched to the drawn rows and columns when read, as `pheatmap` matches them.

Value

What `pheatmap::pheatmap()` returns, invisibly, with the drawn matrix as its `"maidr_matrix"` attribute and, when it has annotations, them as its `"maidr_annotations"` attribute

Attach order

Attach `'pheatmap'` *before* `maidr`, or call `maidr::pheatmap()`:

```
library(pheatmap)
library(maidr)
```

See Also

[show\(\)](#) and [save_html\(\)](#), which read the result.

Examples

```
if (requireNamespace("pheatmap", quietly = TRUE)) {
  m <- matrix(rnorm(20), 4, 5, dimnames = list(letters[1:4], LETTERS[1:5]))
  p <- maidr::pheatmap(m, silent = TRUE)
  maidr::save_html(p, file = tempfile(fileext = ".html"))
}
```

process_facet_panel	<i>Process a single facet panel</i>
---------------------	-------------------------------------

Description

Process a single facet panel

Usage

```
process_facet_panel(
  plot,
  panel_info,
  panel_data,
  facet_groups,
  gtable_panel_name,
  built,
  layout,
  gtable,
  format_config = NULL
)
```

Arguments

plot	The original plot
panel_info	Panel information
panel_data	Panel-specific data
facet_groups	Facet group information
gtable_panel_name	Gtable panel name
built	Built plot data
layout	Layout information
gtable	Gtable object
format_config	Optional format configuration from maidr label functions

Value

Processed panel data

```
process_faceted_plot_data
```

Process a faceted plot and return organized subplot data

Description

Process a faceted plot and return organized subplot data

Usage

```
process_faceted_plot_data(plot, layout, built, gtable, format_config = NULL)
```

Arguments

plot	The faceted ggplot2 object
layout	Layout information
built	Built plot data
gtable	Gtable object
format_config	Optional format configuration from maidr label functions

Value

List with organized subplot data in 2D grid format

process_patchwork_panel

Process a single patchwork panel

Description

Process a single patchwork panel

Usage

```
process_patchwork_panel(
  leaf_plot,
  panel_name,
  panel_index,
  row,
  col,
  layout,
  gtable,
  n_original_layers = NULL
)
```

Arguments

leaf_plot	The leaf ggplot object
panel_name	Panel name from gtable
panel_index	Panel index
row	Panel row
col	Panel column
layout	Layout information
gtable	Gtable object
n_original_layers	Number of layers the user actually wrote. Defaults to every layer of leaf_plot; pass the un-augmented count so injected geoms (violin's boxplot) do not emit a maidr layer of their own.

Value

Processed panel data

process_patchwork_plot_data

Process a patchwork plot and return organized subplot data

Description

Process a patchwork plot and return organized subplot data

Usage

```
process_patchwork_plot_data(plot, layout, gtable, original_plot = NULL)
```

Arguments

plot	The patchwork plot object, with leaves already augmented
layout	Layout information
gtable	Gtable object
original_plot	The un-augmented composition. Supplied so each leaf is processed for the layers the user wrote rather than for the extra geoms a processor injected to render its selectors.

Value

List with organized subplot data in 2D grid format

render_maidr

Render MAIDR Plot in Shiny Server

Description

Creates a Shiny render function for MAIDR widgets using htmlwidgets. This provides automatic dependency injection and robust JavaScript initialization.

Usage

```
render_maidr(
  expr,
  env = parent.frame(),
  quoted = FALSE,
  fig_width = NULL,
  fig_height = NULL,
  hover_mode = NULL
)
```

Arguments

expr	An expression that draws a plot. Either a ggplot object or a lattice (trellis) object, returned rather than printed, or Base R plotting calls – their return values differ (plot() returns NULL, barplot() returns bar midpoints) and are ignored; what counts is whether the expression drew. An expression that draws nothing and returns NULL renders nothing, per Shiny convention.
env	The environment in which to evaluate expr
quoted	Is expr a quoted expression
fig_width, fig_height	The size to draw the chart at, in inches, as width and height set it in show() : each a single positive number no larger than 50, or NULL (the default) for 7 x 5 in, 12 x 6 in for a candlestick chart. They are not the size of the output on the page, which maidr_output() 's width and height set: a chart wider than its output shrinks to fit it. Nothing sizes the chart to its output; the size is the one given here.
hover_mode	How the pointer moves the reader through the chart, as in show() : "pointermove", "click" or "off", or NULL (the default) for <code>getOption("maidr.hover_mode")</code> , and for the reader's own setting when that is unset.

Value

A Shiny render function for use in server

Examples

```
if (interactive()) {
  library(shiny)
  library(ggplot2)
  server <- function(input, output) {
    output$myplot <- render_maidr({
      ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
        geom_bar(stat = "identity")
    })

    # Drawn 10 inches wide and 4 high
    output$wide <- render_maidr(
      {
        ggplot(mtcars, aes(x = wt, y = mpg)) +
          geom_point()
      },
      fig_width = 10,
      fig_height = 4
    )

    # Only a click moves the reader through this one
    output$quiet <- render_maidr(
      {
        ggplot(mtcars, aes(x = wt, y = mpg)) +
          geom_point()
      }
    )
  }
}
```

```
    },
    hover_mode = "click"
  )
}
```

run_example	<i>Run MAIDR Example Plots</i>
-------------	--------------------------------

Description

Launches example plots demonstrating MAIDR’s accessible visualization capabilities. Each example creates an interactive plot using `show()`.

Usage

```
run_example(example = NULL, type = c("ggplot2", "base_r", "lattice"))
```

Arguments

example	Character string specifying which example to run. If NULL (the default), lists all available examples.
type	Character string specifying the plot system to use: "ggplot2" (default), "base_r" or "lattice".

Details

Available examples include various plot types such as bar charts, histograms, scatter plots, line plots, boxplots, heatmaps, and more.

The "lattice" examples are a bar chart [experimental], a histogram [experimental], a scatter plot [experimental], a box plot [experimental] and a conditioned (faceted) scatter plot [experimental]. Every chart maidr reads from 'lattice' is experimental: none has been through a user study, and each may change without a deprecation period. They need the 'lattice' package.

Each example script creates a plot and calls `show()` to display it in your default web browser with full MAIDR accessibility features including keyboard navigation and screen reader support.

Value

Invisibly returns NULL. Called for its side effect of displaying an interactive plot in the browser or listing available examples.

See Also

[show\(\)](#) for displaying plots, [save_html\(\)](#) for saving to file

Examples

```
# List all available examples
run_example()

if (interactive()) {
  # Run ggplot2 bar chart example
  run_example("bar")

  # Run Base R histogram example
  run_example("histogram", type = "base_r")

  # Run lattice box plot example [experimental]
  run_example("boxplot", type = "lattice")
}
```

save_html

Save Interactive Plot as HTML File

Description

Save a ggplot2, lattice or Base R plot as an HTML file with interactive MAIDR accessibility features.

Usage

```
save_html(
  plot = NULL,
  file = "plot.html",
  use_cdn = NULL,
  width = NULL,
  height = NULL,
  hover_mode = NULL,
  ...
)
```

Arguments

plot	A ggplot2 object, a lattice (trellis) object, the object <code>pheatmap::pheatmap()</code> returns (experimental), a 'ComplexHeatmap' Heatmap or HeatmapList (experimental), or NULL for Base R auto-detection
file	File path where to save the HTML file (e.g., "plot.html")
use_cdn	Logical. Controls where MAIDR.js is loaded from: • TRUE: Use CDN. The file is self-contained but needs internet access when it is viewed. It names the latest published MAIDR.js by version (looked up once per R session, or the bundled version when the lookup cannot be made); pin a version with <code>options(maidr.cdn_version = ...)</code>, see <code>?"maidr-options"</code>.

	• FALSE or NULL (default): Use the bundled files. The MAIDR.js library is written to a lib/ folder beside file, which has to travel with it.
width, height	The size to draw the chart at, in inches: each a single positive number no larger than 50, or NULL (the default) for 7 x 5 in, 12 x 6 in for a candlestick chart. A side not given takes its default. See Chart size .
hover_mode	How the pointer moves the reader through the chart: "pointermove" (hovering moves the reader's position and its highlight), "click" (only a click moves it) or "off" (the pointer is ignored, and only the keyboard moves it). It is the chart's starting value for the reader's Hover Mode setting: a reader who has changed that setting keeps theirs. NULL (the default) takes <code>getOption("maidr.hover_mode")</code> , and when that is unset writes nothing, so <code>maidr.js</code> uses the reader's setting, whose default is "pointermove". Read by <code>maidr.js</code> 4.15.0 and later, the version bundled with this package; an older one, which <code>maidr.cdn_version</code> can pin, ignores it. See <code>? "maidr-options"</code> .
...	Additional arguments passed to internal functions

Details

By default the MAIDR.js library is written to a lib/ folder beside file, and the two have to be shared together: zip the folder that holds both, or copy both. An .html sent on its own loads no MAIDR.js and shows a plain, inaccessible chart. `use_cdn = TRUE` writes one self-contained file instead, which needs internet access whenever it is viewed and loads the latest published MAIDR.js from jsDelivr rather than the copy bundled with this package.

Value

The file path where the HTML was saved (invisibly)

Base R charts

A Base R chart is read from the plotting calls recorded on the current device, and is the page that device shows: the last one. R draws a high-level plot on a new page when it moves past the last panel of the page it is on – every plot on a page of one panel, the fifth under `par(mfrow = c(2, 2))`, the first after `par(mfrow = )`, `par(mfcol = )` or `layout()` sets a page up again – and `plot.new()` and `frame()` move on a panel as a plot does. So after `hist(a)`; `hist(b)` the chart is the histogram of b alone, and after five plots under `par(mfrow = c(2, 2))` it is the fifth, in the first panel of a 2 x 2 grid. Each plot of a grid is in the panel R drew it in: a plot drawn after `par(new = TRUE)`, or with `add = TRUE`, is drawn in the panel of the plot before it, one `par(mfg = )` sends to a panel out of turn is in that panel, and a panel `plot.new()` or `frame()` passed over stays empty. So it is where the call that set the grid up was not recorded: one made through `graphics::par()` or `graphics::layout()`, before `maidr_on()`, or before an earlier `show()` or `save_html()` on the device. A plot drawn in a region of the page `par(fig = )` gave it, as an inset is, or in a screen of `split.screen()`, is drawn in that region, and read with the plot before it. `lines()`, `points()`, `abline()`, `text()`, `legend()`, `title()`, `axis()` and the other low-level calls add to the plot they are drawn on: after `par(mfg = )`, or `screen(n, new = FALSE)` of `split.screen()`, sends R back to the panel or screen of an earlier plot, to that plot. Drawn on a panel `plot.new()` or `frame()` took, as a legend of its own is, or on a plot `maidr` does not record, such as `smoothScatter()`, they are drawn there, and read as part of no plot. Drawn over a plot after `par(new = TRUE)` and `plot.new()`, as a second series with an axis of

its own is – also over the plot of an earlier panel or screen `par(mfg = )` or `screen()` sent R back to – they are drawn in the coordinates they were drawn in, and read with that plot. Nothing drawn on an earlier page reaches the chart: not its data, its titles or the size it is drawn at. A page `replayPlot()` puts back, from a plot `recordPlot()` saved on a device that keeps a display list, is the page the chart is read from, with what was drawn on it when it was saved and what has been drawn on it since. A page that holds no plot `maidr` recorded – one R started with `plot.new()` or `frame()`, with a plot `maidr` does not record, or with one drawn while `maidr_off()` was in effect, even with `lines()` or `text()` drawn on it since, or one `replayPlot()` put back from such a page, or from a plot saved in another session or on another device – is not read as the plot before it: `show()` and `save_html()` stop, and say so. Each figure of an R Markdown or Quarto document is read the same way, from the calls on its own page.

Chart size

A chart is drawn at a size in inches, as `ggplot2::ggsave()` and knitr's `fig.width` and `fig.height` size a figure, and its SVG is 72 pixels to the inch: a 10 x 4 in chart is 720 pixels wide and 288 high (a knitted page shows it at the size of its chunk's figure instead: see `maidr_on()`). The size is the room the chart is laid out in – how far apart its ticks and labels are, how `lattice` arranges its panels, where Base R puts its titles – and not what `maidr` reads out: the data, titles and axis labels a reader hears are the same at every size. Only how a reader moves between panels can change: `lattice` arranges the panels of a chart conditioned on one variable with no `layout` = for the shape of the page, 1 x 2 at 7 x 5 in and 2 x 1 at 5 x 8 in for two panels, and the subplot grid a reader moves through follows; give `layout` = to keep it. The size is set by width and height in `show()` and `save_html()`, by `fig_width` and `fig_height` in `render_maidr()`, and by the chunk's `fig.width` and `fig.height` in an R Markdown or Quarto document (see `maidr_on()`). Nothing else sets it: not the size of the device a chart was drawn on, nor the size of the window, viewer or Shiny output it is shown in, which shrinks a chart wider than itself to fit.

Unset, a chart is 7 x 5 in. A candlestick chart is 12 x 6 in, and is never drawn smaller than that: `quantmod::chartSeries()` needs the room for its title, date range and date labels. A smaller size asked for is enlarged, with a message naming the size the chart is drawn at. A side larger than 50 in is refused, as `ggplot2::ggsave()` refuses one: the size is in inches, not the pixels `maidr_output()` takes.

A Base R chart's margins and text take the same room at every size, as do the cells a `layout()` call sizes with `lcm()`, so a chart can be too small for them – R itself stops with "figure margins too large", or for those cells "figure region too large", at such a size. `maidr` draws the chart with the margins and text size its `par()` calls set, so it fits where R draws it with them. Asked for a size too small, `maidr` stops too, with an error naming the size, rather than show an empty chart: draw it larger. With no size asked for, a chart too small for 7 x 5 in, such as a `par(mfrow)` grid of five rows or more, is drawn on the 7 x 7 in page `maidr` laid every Base R chart out on before it drew one at its size. Where that is too small too, it is drawn on the smallest larger page that leaves each of its plots a sixth of an inch, 12 px, each way, each side grown in whole inches only as far as it needs: 7 x 9 in for six rows, 12 x 5 in for twelve columns. A message names the size it is drawn at. The picture of a Base R chart `maidr` cannot read is held to its size in the same way.

Examples

```
# ggplot2 bar chart
library(ggplot2)
p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
```

```

    geom_bar(stat = "identity")

maidr::save_html(p, tempfile(fileext = ".html"))

# The same chart, 10 inches wide and 4 high
maidr::save_html(p, tempfile(fileext = ".html"), width = 10, height = 4)

# ggplot2 violin plot
p_violin <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_violin(fill = "lightblue", alpha = 0.7) +
  labs(title = "MPG by Cylinders", x = "Cylinders", y = "MPG")

maidr::save_html(p_violin, tempfile(fileext = ".html"))

# lattice chart [experimental]

if (requireNamespace("lattice", quietly = TRUE)) {
  p_lattice <- lattice::bwplot(factor(cyl) ~ mpg, data = mtcars)
  maidr::save_html(p_lattice, tempfile(fileext = ".html"))
}

# Base R example (requires interactive session for function patching)
if (interactive()) {
  barplot(c(10, 20, 30), names.arg = c("A", "B", "C"))
  maidr::save_html(file = tempfile(fileext = ".html"))
}

```

show

Display Interactive MAIDR Plot

Description

Display a ggplot2, lattice or Base R plot as an interactive, accessible visualization using the MAIDR (Multimodal Access and Interactive Data Representation) system.

Usage

```

show(
  plot = NULL,
  use_cdn = NULL,
  shiny = FALSE,
  as_widget = FALSE,
  width = NULL,
  height = NULL,
  hover_mode = NULL,
  ...
)

```

Arguments

plot	A ggplot2 object, a lattice (trellis) object, the object <code>pheatmap::pheatmap()</code> returns (experimental), a 'ComplexHeatmap' Heatmap or HeatmapList (experimental), or NULL for Base R auto-detection
use_cdn	Logical. Controls where MAIDR.js is loaded from: • TRUE: Use the jsDelivr CDN (requires internet), which loads the latest published MAIDR.js rather than the bundled copy. The version is looked up once per R session; pin one with <code>options(maidr.cdn_version = ...)</code>, see <code>?"maidr-options"</code>. • FALSE: Use local bundled files (works offline, and makes no network request) • NULL (default): Use the bundled files, so the viewer works offline. With <code>as_widget = TRUE</code> the widget instead auto-detects internet availability and uses the CDN when online, as the Shiny path does.
shiny	If TRUE, returns just the SVG content instead of full HTML document
as_widget	If TRUE, returns an <code>htmlwidget</code> object instead of opening in browser
width, height	The size to draw the chart at, in inches: each a single positive number no larger than 50, or NULL (the default) for 7 x 5 in, 12 x 6 in for a candlestick chart. A side not given takes its default. With <code>as_widget = TRUE</code> they size the chart in the widget, not the widget, whose own CSS size is set on the widget returned: <code>widget\$width <- "300px"</code> . See Chart size .
hover_mode	How the pointer moves the reader through the chart: <code>"pointermove"</code> (hovering moves the reader's position and its highlight), <code>"click"</code> (only a click moves it) or <code>"off"</code> (the pointer is ignored, and only the keyboard moves it). It is the chart's starting value for the reader's Hover Mode setting: a reader who has changed that setting keeps theirs. NULL (the default) takes <code>getOption("maidr.hover_mode")</code> , and when that is unset writes nothing, so <code>maidr.js</code> uses the reader's setting, whose default is <code>"pointermove"</code> . Read by <code>maidr.js</code> 4.15.0 and later, the version bundled with this package; an older one, which <code>maidr.cdn_version</code> can pin, ignores it. See <code>?"maidr-options"</code> .
...	Additional arguments passed to internal functions

Details

Attaching `maidr` masks `methods::show()`. An object that is not a plot `maidr` renders – an S4 object, a vector, a data frame – is handed to `methods::show()`, so it prints as it did before `maidr` was attached. In a script or a package, call `maidr::show()` and `methods::show()` by name; `?"base-r-wrappers"` lists everything else attaching `maidr` masks.

Under `webR` there is no browser to open, so the chart is added to the page the session runs in: as an `iframe` in the element with id `maidr-output`, or at the end of `<body>` when there is none. A page that defines `globalThis.maidrWebRShow(html)`, or R code that sets `options(maidr.webR_display = function(html) ...)`, receives the document instead. A `plotly`, `highcharter` or `echarts4r` widget is shown the same way, made accessible with `maidr_htmlwidget()` first; its document carries the chart library, a few megabytes. Any other `htmlwidget` is refused, with the message `maidr_htmlwidget()` gives.

Value

Invisible NULL. The plot is displayed in RStudio Viewer or browser as a side effect.

Base R charts

A Base R chart is read from the plotting calls recorded on the current device, and is the page that device shows: the last one. R draws a high-level plot on a new page when it moves past the last panel of the page it is on – every plot on a page of one panel, the fifth under `par(mfrow = c(2, 2))`, the first after `par(mfrow = )`, `par(mfcol = )` or `layout()` sets a page up again – and `plot.new()` and `frame()` move on a panel as a plot does. So after `hist(a); hist(b)` the chart is the histogram of `b` alone, and after five plots under `par(mfrow = c(2, 2))` it is the fifth, in the first panel of a 2 x 2 grid. Each plot of a grid is in the panel R drew it in: a plot drawn after `par(new = TRUE)`, or with `add = TRUE`, is drawn in the panel of the plot before it, one `par(mfg = )` sends to a panel out of turn is in that panel, and a panel `plot.new()` or `frame()` passed over stays empty. So it is where the call that set the grid up was not recorded: one made through `graphics::par()` or `graphics::layout()`, before `maidr_on()`, or before an earlier `show()` or `save_html()` on the device. A plot drawn in a region of the page `par(fig = )` gave it, as an inset is, or in a screen of `split.screen()`, is drawn in that region, and read with the plot before it. `lines()`, `points()`, `abline()`, `text()`, `legend()`, `title()`, `axis()` and the other low-level calls add to the plot they are drawn on: after `par(mfg = )`, or `screen(n, new = FALSE)` of `split.screen()`, sends R back to the panel or screen of an earlier plot, to that plot. Drawn on a panel `plot.new()` or `frame()` took, as a legend of its own is, or on a plot `maidr` does not record, such as `smoothScatter()`, they are drawn there, and read as part of no plot. Drawn over a plot after `par(new = TRUE)` and `plot.new()`, as a second series with an axis of its own is – also over the plot of an earlier panel or screen `par(mfg = )` or `screen()` sent R back to – they are drawn in the coordinates they were drawn in, and read with that plot. Nothing drawn on an earlier page reaches the chart: not its data, its titles or the size it is drawn at. A page `replayPlot()` puts back, from a plot `recordPlot()` saved on a device that keeps a display list, is the page the chart is read from, with what was drawn on it when it was saved and what has been drawn on it since. A page that holds no plot `maidr` recorded – one R started with `plot.new()` or `frame()`, with a plot `maidr` does not record, or with one drawn while `maidr_off()` was in effect, even with `lines()` or `text()` drawn on it since, or one `replayPlot()` put back from such a page, or from a plot saved in another session or on another device – is not read as the plot before it: `show()` and `save_html()` stop, and say so. Each figure of an R Markdown or Quarto document is read the same way, from the calls on its own page.

Chart size

A chart is drawn at a size in inches, as `ggplot2::ggsave()` and knitr's `fig.width` and `fig.height` size a figure, and its SVG is 72 pixels to the inch: a 10 x 4 in chart is 720 pixels wide and 288 high (a knitted page shows it at the size of its chunk's figure instead: see `maidr_on()`). The size is the room the chart is laid out in – how far apart its ticks and labels are, how lattice arranges its panels, where Base R puts its titles – and not what `maidr` reads out: the data, titles and axis labels a reader hears are the same at every size. Only how a reader moves between panels can change: lattice arranges the panels of a chart conditioned on one variable with no `layout =` for the shape of the page, 1 x 2 at 7 x 5 in and 2 x 1 at 5 x 8 in for two panels, and the subplot grid a reader moves through follows; give `layout =` to keep it. The size is set by width and height in `show()` and `save_html()`, by `fig_width` and `fig_height` in `render_maidr()`, and by the chunk's `fig.width` and `fig.height` in an R Markdown or Quarto document (see `maidr_on()`). Nothing else sets it:

not the size of the device a chart was drawn on, nor the size of the window, viewer or Shiny output it is shown in, which shrinks a chart wider than itself to fit.

Unset, a chart is 7 x 5 in. A candlestick chart is 12 x 6 in, and is never drawn smaller than that: `quantmod::chartSeries()` needs the room for its title, date range and date labels. A smaller size asked for is enlarged, with a message naming the size the chart is drawn at. A side larger than 50 in is refused, as `ggplot2::ggsave()` refuses one: the size is in inches, not the pixels `maidr::output()` takes.

A Base R chart's margins and text take the same room at every size, as do the cells a `layout()` call sizes with `lcm()`, so a chart can be too small for them – R itself stops with "figure margins too large", or for those cells "figure region too large", at such a size. `maidr` draws the chart with the margins and text size its `par()` calls set, so it fits where R draws it with them. Asked for a size too small, `maidr` stops too, with an error naming the size, rather than show an empty chart: draw it larger. With no size asked for, a chart too small for 7 x 5 in, such as a `par(mfrow)` grid of five rows or more, is drawn on the 7 x 7 in page `maidr` laid every Base R chart out on before it drew one at its size. Where that is too small too, it is drawn on the smallest larger page that leaves each of its plots a sixth of an inch, 12 px, each way, each side grown in whole inches only as far as it needs: 7 x 9 in for six rows, 12 x 5 in for twelve columns. A message names the size it is drawn at. The picture of a Base R chart `maidr` cannot read is held to its size in the same way.

Examples

```
# ggplot2 bar chart
library(ggplot2)
p <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_bar(stat = "identity")

maidr::show(p)

# The same chart, 10 inches wide and 4 high
maidr::show(p, width = 10, height = 4)

# ggplot2 violin plot
p_violin <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_violin(fill = "lightblue", alpha = 0.7) +
  labs(title = "MPG by Cylinders", x = "Cylinders", y = "MPG")

maidr::show(p_violin)

# lattice chart [experimental]

if (requireNamespace("lattice", quietly = TRUE)) {
  maidr::show(lattice::xyplot(mpg ~ wt, data = mtcars))
}

# Hovering does not move the reader; a click does

maidr::show(p, hover_mode = "click")
```

```
# Base R example (requires interactive session for function patching)
if (interactive()) {
  barplot(c(10, 20, 30), names.arg = c("A", "B", "C"))
  maidr::show(width = 6, height = 4)
}
```

Index

abline (base-r-wrappers), 3
acf (base-r-wrappers), 3
arrows (base-r-wrappers), 3
assocplot (base-r-wrappers), 3
axis (base-r-wrappers), 3

barplot (base-r-wrappers), 3
base-r-wrappers, 3
biplot (base-r-wrappers), 3
boxplot (base-r-wrappers), 3
bxp (base-r-wrappers), 3

ccf (base-r-wrappers), 3
cdplot (base-r-wrappers), 3
chartSeries (base-r-wrappers), 3
combine_facet_layer_data, 7
combine_facet_layer_selectors, 7
contour (base-r-wrappers), 3
coplot (base-r-wrappers), 3
cpgram (base-r-wrappers), 3
curve (base-r-wrappers), 3

dotchart (base-r-wrappers), 3

extract_leaf_plot_layout, 8
extract_patchwork_leaves, 8

filled.contour (base-r-wrappers), 3
find_children_by_type, 9
find_graphics_plot_grob, 9
find_patchwork_panels, 10
find_patchwork_panels(), 8
fourfoldplot (base-r-wrappers), 3

generate_robust_css_selector, 10
generate_robust_selector, 11
get_facet_groups, 11
ggplot2::geom_path(), 29, 31
ggplot2::geom_point(), 23, 35, 37, 38
ggplot2::geom_rect(), 17
ggplot2::geom_step(), 35

ggplot2::ggsave(), 49, 52, 53
grid (base-r-wrappers), 3

heatmap (base-r-wrappers), 3
hist (base-r-wrappers), 3

image (base-r-wrappers), 3
interaction.plot (base-r-wrappers), 3

lag.plot (base-r-wrappers), 3
layout (base-r-wrappers), 3
legend (base-r-wrappers), 3
lines (base-r-wrappers), 3

maidr-options, 12, 16, 26
maidr_download_dotpad_sdk, 13, 14, 15
maidr_gantt, 17
maidr_gantt(), 32
maidr_get_fallback, 19
maidr_get_fallback(), 34
maidr_htmlwidget, 20, 51
maidr_manhattan, 22
maidr_manhattan(), 38
maidr_off, 24
maidr_off(), 3, 7, 25, 27, 49, 52
maidr_on, 25
maidr_on(), 7, 25, 48, 49, 52
maidr_output, 28, 45
maidr_output(), 49, 53
maidr_pr_curve, 28
maidr_roc, 31
maidr_roc(), 30, 36
maidr_set_fallback, 33
maidr_set_fallback(), 19
maidr_survival, 34
maidr_volcano, 37
maidr_volcano(), 24, 36
map_visual_to_dom_panel, 39
matplot (base-r-wrappers), 3
monthplot (base-r-wrappers), 3

mosaicplot (base-r-wrappers), 3
mtext (base-r-wrappers), 3
organize_facet_grid, 40
pacf (base-r-wrappers), 3
pairs (base-r-wrappers), 3
par (base-r-wrappers), 3
persp (base-r-wrappers), 3
pheatmap, 40
pheatmap(), 6
pie (base-r-wrappers), 3
plot (base-r-wrappers), 3
points (base-r-wrappers), 3
polygon (base-r-wrappers), 3
pROC::auc(), 31
pROC::ggroc(), 32
process_facet_panel, 41
process_faceted_plot_data, 42
process_patchwork_panel, 43
process_patchwork_plot_data, 44
qqline (base-r-wrappers), 3
qqnorm (base-r-wrappers), 3
qqplot (base-r-wrappers), 3
rect (base-r-wrappers), 3
render_mair, 44
render_mair(), 49, 52
run_example, 46
save_html, 47
save_html(), 3, 7, 16, 18, 24, 30, 32, 36, 38, 40, 41, 46, 49, 52
segments (base-r-wrappers), 3
show, 21, 45, 50
show(), 3, 6, 7, 16, 18, 24–26, 30, 32, 36, 38, 40, 41, 46, 49, 52
spectrum (base-r-wrappers), 3
spineplot (base-r-wrappers), 3
split.screen (base-r-wrappers), 3
stars (base-r-wrappers), 3
stem (base-r-wrappers), 3
stripchart (base-r-wrappers), 3
sunflowerplot (base-r-wrappers), 3
symbols (base-r-wrappers), 3
termplot (base-r-wrappers), 3
text (base-r-wrappers), 3
title (base-r-wrappers), 3
vioplot (base-r-wrappers), 3
wordcloud (base-r-wrappers), 3
yardstick::pr_curve(), 30
yardstick::roc_curve(), 32