

Package ‘maidr’

September 29, 2026

Type Package

Title Multimodal Access and Interactive Data Representation

Version 0.5.0

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 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, htmltools, htmlwidgets, jsonlite, methods, shiny, stats, svglite (>= 2.1.1), xml2, rlang, R6

Suggests testthat (>= 3.0.0), lintr, styler, goodpractice, cyclocomp, knitr, rmarkdown, quarto, patchwork, scales, tibble, tidyquant, quantmod, TTR, zoo, xts, pkgload, hexbin, vioplot, wordcloud, sm, gridGraphics, pROC, yardstick, plotly, highcharter, echarts4r

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author JooYoung Seo [aut, cph, cre],

Niranjan Kalaiselvan [aut]

Maintainer JooYoung Seo <jseo1005@illinois.edu>

Repository CRAN

Date/Publication 2026-09-29 21:20:22 UTC

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	14
maidr_gantt	16
maidr_get_fallback	18
maidr_htmlwidget	19
maidr_off	20
maidr_on	21
maidr_output	22
maidr_roc	22
maidr_set_fallback	25
map_visual_to_dom_panel	26
organize_facet_grid	26
process_faceted_plot_data	27
process_facet_panel	27
process_patchwork_panel	28
process_patchwork_plot_data	29
render_maidr	30
run_example	31
save_html	32
show	33

Index

35

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	<code>axis()</code> 's own arguments, which its wrapper names so that the tick labels a chart is given can be recorded; passed on to <code>graphics::axis()</code> 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.

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

`maidr`'s `show()` takes a `ggplot2` 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.

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

panel_info	Panel information from layout
built	Built plot data

Value

List of facet group information

maidr-options	<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. 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. Default: TRUE.

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

Setting Options

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

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

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

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

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, knitr and Shiny paths render their charts in `srcdoc` frames, where a relative path has nothing to resolve against, 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, knitr 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.

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
```

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 knitr document, whose charts live in `srcdoc` frames with no base URL for a relative path to resolve against – 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

Declare that a rectangle layer draws a schedule

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> .

...	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 "identity"; 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.
<code>lane_axis</code>	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.
<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 `+`.

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	<i>Make a plotly, highcharter or echarts4r htmlwidget accessible</i>
------------------	--

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)
```

Arguments

<code>widget</code>	An htmlwidget created by plotly, highcharter or echarts4r.
<code>use_cdn</code>	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>.

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.

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()

## End(Not run)
```

maidr_off

Disable MAIDR Plot Interception

Description

Disables automatic MAIDR rendering and restores normal plot behavior. After calling this, Base R plots display in the standard graphics window and `ggplot2` objects render with the default `ggplot2` method.

Usage

```
maidr_off()
```

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 and Base R plots, and installs the knitr hooks that an R Markdown or Quarto document needs.

Usage

```
maidr_on()
```

Details

Interception is on by default after `library(maidr)`: printing a ggplot2 object opens it in the MAIDR viewer, and Base R plotting calls are recorded until [show\(\)](#) is called. Calling `maidr_on()` yourself is needed in two places: after [maidr_off\(\)](#), to start again, and once in the setup chunk of an R Markdown or Quarto document, where it registers the `knit_print` methods and the plot hook that turn every plot the document draws into an accessible chart. `library(maidr)` alone installs neither.

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 widgets
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_roc	<i>Declare that a path layer draws a ROC curve</i>
-----------	--

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

mapping	Aesthetics, as for <code>ggplot2::geom_path()</code> : x (the false positive rate) and y (the true positive rate) 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": a declared curve is always drawn from the author's own rates.
auc	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.
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 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)
```

map_visual_to_dom_panel	<i>Map visual panel position to DOM panel name</i>
-------------------------	--

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

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_facet_panel *Process a single facet panel*

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

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

Value

Processed panel data

`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

<code>leaf_plot</code>	The leaf ggplot object
<code>panel_name</code>	Panel name from gtable
<code>panel_index</code>	Panel index
<code>row</code>	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	<i>Render MAIDR Plot in Shiny Server</i>
--------------	--

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)
```

Arguments

expr	An expression that draws a plot. Either a ggplot object, 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

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")
    })
  }
}
```

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"))
```

Arguments

example	Character string specifying which example to run. If <code>NULL</code> (the default), lists all available examples.
type	Character string specifying the plot system to use. Either <code>"ggplot2"</code> (default) or <code>"base_r"</code> .

Details

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

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")
}
```

save_html

*Save Interactive Plot as HTML File***Description**

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

Usage

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

Arguments

plot	A ggplot2 object 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 <code>lib/</code> folder beside <code>file</code>, which has to travel with it.
...	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)

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"))
```

```
# 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"))

# 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 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, ...)
```

Arguments

plot	A ggplot2 object 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 knitr and Shiny paths do.
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
...	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.

Value

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

Examples

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

maidr::show(p)

# 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)

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

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(), 23
ggplot2::geom_rect(), 16, 17
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, 15
maidr_download_dotpad_sdk, 12, 13, 14
maidr_gantt, 16
maidr_gantt(), 24
maidr_get_fallback, 18
maidr_get_fallback(), 25
maidr_htmlwidget, 19
maidr_off, 20
maidr_off(), 3, 6, 21
maidr_on, 21
maidr_on(), 6, 21
maidr_output, 22
maidr_roc, 22
maidr_set_fallback, 25
maidr_set_fallback(), 19
map_visual_to_dom_panel, 26
matplot (base-r-wrappers), 3
monthplot (base-r-wrappers), 3
mosaicplot (base-r-wrappers), 3
mtext (base-r-wrappers), 3

organize_facet_grid, 26

pacf (base-r-wrappers), 3
pairs (base-r-wrappers), 3
par (base-r-wrappers), 3
persp (base-r-wrappers), 3
pie (base-r-wrappers), 3
plot (base-r-wrappers), 3
points (base-r-wrappers), 3

`polygon` (base-r-wrappers), 3
`pROC::auc()`, 23
`pROC::ggroc()`, 24
`process_facet_panel`, 27
`process_faceted_plot_data`, 27
`process_patchwork_panel`, 28
`process_patchwork_plot_data`, 29

`qqline` (base-r-wrappers), 3
`qqnorm` (base-r-wrappers), 3
`qqplot` (base-r-wrappers), 3

`rect` (base-r-wrappers), 3
`render_maidr`, 30
`run_example`, 31

`save_html`, 32
`save_html()`, 3, 6, 15, 18, 24, 31
`segments` (base-r-wrappers), 3
`show`, 33
`show()`, 3, 6, 15, 18, 21, 24, 31
`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::roc_curve()`, 24