Package 'rpic'

November 2, 2026

Title The 'pic' Graphics Language Rendered to SVG, PNG and PDF

Version 0.12.1

Description Renders diagrams written in the 'pic' picture-drawing language (Kernighan, 1984) to SVG, PNG and PDF, using 'rpic' <https://github.com/milkway/rpic-lang>, a reimplementation of 'pic' in 'Rust' with no system dependencies. Includes a native circuit-element library in the spirit of 'circuit_macros', TeX math labels typeset natively, structured compile diagnostics, and a 'knitr' language engine for inline diagrams in 'R Markdown' and 'Quarto' documents.

License BSD_2_clause + file LICENSE

Encoding UTF-8

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.88.0)

Suggests jsonlite, knitr, rmarkdown

VignetteBuilder knitr

URL https://milkway.github.io/rpic-r/, https://github.com/milkway/rpic-r, https://github.com/milkway/rpic-lang

BugReports https://github.com/milkway/rpic-r/issues

Config/rextendr/version 0.5.0

Depends R (>= 4.2)

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author André Leite [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-4718-9766>), The authors of the dependency Rust crates [cph] (see inst/AUTHORS for details)

Maintainer André Leite <leite@castlab.org>

Repository CRAN

Date/Publication 2026-11-02 20:30:02 UTC

Contents

rpic_knitr_engine 2
rpic_manifest 3
rpic_pdf 3
rpic_png 4
rpic_register_knitr 4
rpic_scene 5
rpic_svg 6

Index **7**

`rpic_knitr_engine` *knitr engine for* `{rpic}` *chunks*

Description

Register with `rpic_register_knitr()`, then write pic code in a chunk:

```
```{rpic, circuits=TRUE}
A:(0,0); B:(2,0)
resistor(A,B)
```
```

Chunk options: `circuits`, `texlabels`, `scale`.

Usage

```
rpic_knitr_engine(options)
```

Arguments

`options`	knitr chunk options.

Value

The chunk output produced by `knitr::engine_output()` (the rendered diagram as an included figure).

rpic_manifest *Compile to a JSON bundle (as a string)*

Description

Returns `{svg, animations, diagnostics, warnings, objects}` — `warnings` carries structured compiler warnings for accepted-but-ignored input (unknown attribute words, unknown `animate` effects), each with the same fields as the `rpic_error` diagnostic. `objects` lists every drawn object: its SVG `id` (`"s0"`, `"s1"`, ...), `kind`, bounding box in viewBox units and source position. An `interactions` array is added when the source marks objects `draggable`. On a pic error the JSON is `{error, error_info}` instead (no condition is raised — the error travels in-band).

Usage

```
rpic_manifest(src, circuits = FALSE, texlabels = FALSE)
```

Arguments

`src`	pic source code.
`circuits`	load the native circuit-element library (or write `copy "circuits"` in the source itself).
`texlabels`	typeset fully `$...$`-delimited labels as TeX math (initializer only — the source can still set `texlabels = 0`).

Value

a JSON string; parse with e.g. `jsonlite::fromJSON()`.

Examples

```
rpic_manifest('box; animate last box with "pop"')
```

rpic_pdf *Render pic source to a PDF file*

Description

Render pic source to a PDF file

Usage

```
rpic_pdf(src, file, circuits = FALSE, texlabels = FALSE)
```

Arguments

`src`	pic source code.
`file`	output path.
`circuits`	load the native circuit-element library.
`texlabels`	typeset `$...$` labels as TeX math.

Value

the file path, invisibly.

`rpic_png` *Render pic source to a PNG file*

Description

Render pic source to a PNG file

Usage

```
rpic_png(src, file, scale = 2, circuits = FALSE, texlabels = FALSE)
```

Arguments

`src`	pic source code.
`file`	output path.
`scale`	raster scale (1 = 96 dpi).
`circuits`	load the native circuit-element library.
`texlabels`	typeset `$...$` labels as TeX math.

Value

the file path, invisibly.

`rpic_register_knitr` *Register the rpic knitr engine*

Description

Register the rpic knitr engine

Usage

```
rpic_register_knitr()
```

Value

No return value; called for its side effect of registering the `rpic` engine with knitr::knit_engines.

`rpic_scene` *Compile to the scene JSON (the resolved draw list)*

Description

The scene is the engine's device-independent output (format 1, see `docs/scene.md` in the engine repository): the fully resolved draw list — final geometry in inches (y-up), curves as cubic Béziers, style as descriptors with colors resolved to hex, text as text with baseline anchor points, every variable's final value under `vars`, and items in draw order carrying the same stable `s<N>` ids as the SVG. Use it to build your own backend (maps, TikZ, editors) on top of rpic's layout.

Usage

```
rpic_scene(
  src,
  flatten = NULL,
  outline = FALSE,
  circuits = FALSE,
  texlabels = FALSE
)
```

Arguments

`src`	pic source code.
`flatten`	optional tolerance in inches: deliver every curve as a polyline within this distance of the exact cubics (`NULL` keeps the Béziers).
`outline`	embed typeset TeX math as self-contained SVG glyph fragments (`outline` keys) instead of metrics-only entries.
`circuits`	load the native circuit-element library (or write `copy "circuits"` in the source itself).
`texlabels`	typeset fully `$...$`-delimited labels as TeX math (initializer only — the source can still set `texlabels = 0`).

Details

Anything the SVG paints that the scene cannot represent is reported as an R `warning()` (engine kind `scene_unsupported`) — never silently dropped.

Value

a JSON string; parse with e.g. `jsonlite::fromJSON()`.

Errors

Compile errors raise the same classed `rpic_error` condition as `rpic_svg()`.

Examples

```
scene <- rpic_scene('geo_lon = -34.88
B: box "hub"
arrow right from B.e')
if (requireNamespace("jsonlite", quietly = TRUE)) {
  s <- jsonlite::fromJSON(scene, simplifyVector = FALSE)
  s$vars$geo_lon      # -34.88
  s$items[[2]]$kind   # "arrow"
}
```

`rpic_svg` *Render pic source to an SVG string*

Description

Render pic source to an SVG string

Usage

```
rpic_svg(src, circuits = FALSE, texlabels = FALSE)
```

Arguments

`src`	pic source code.
`circuits`	load the native circuit-element library (or write `copy "circuits"` in the source itself).
`texlabels`	typeset fully `$...$`-delimited labels as TeX math (initializer only — the source can still set `texlabels = 0`).

Value

an SVG string.

Errors

Compile errors are raised as a classed `rpic_error` condition whose `info` field holds the structured diagnostic (`message`, `line`, `col`, `end_col`, `file`, `kind`, `found`, `expected`, `hint`; absent values are `NA`, and `file` names a `copy` include — `NA` means your own input). Positions are relative to your own source, even with `circuits = TRUE`:

```
tryCatch(
  rpic_svg("bxo", circuits = TRUE),
  rpic_error = function(e) e$info$hint  # "did you mean `box`?"
)
```

Examples

```
rpic_svg('box "hi"; arrow; circle "x"')
tryCatch(rpic_svg("bxo"), rpic_error = function(e) e$info$line)
```

Index

knitr::engine_output(), *2*
knitr::knit_engines, *4*

rpic_knitr_engine, 2
rpic_manifest, 3
rpic_pdf, 3
rpic_png, 4
rpic_register_knitr, 4
rpic_register_knitr(), *2*
rpic_scene, 5
rpic_svg, 6
rpic_svg(), *5*