Package ‘cardargus’

February 5, 2026

Title Generate SVG Information Cards with Embedded Fonts and Badges

Version 0.2.1

Description Create self-contained SVG information cards with embedded 'Google Fonts', shields-style badges, and custom logos. Cards are fully portable SVG files ideal for dashboards, reports, and web applications. Includes functions to export cards to PNG format and display them in 'R Markdown' and 'Quarto' documents.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

URL `https://github.com/StrategicProjects/cardargus`

BugReports `https://github.com/StrategicProjects/cardargus/issues`

Imports glue, cli, digest, gdtools, magick, rsvg

Suggests base64enc, chromote, curl, gfonts, htmltools, knitr, rmarkdown, showtext, sysfonts, systemfonts, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Andre Leite [aut, cre],
Hugo Vasconcelos [aut],
Diogo Bezerra [aut]

Maintainer Andre Leite <`leite@castlab.org`>

Repository CRAN

Date/Publication 2026-02-05 12:10:10 UTC

Contents

batch_svg_to_png *Batch convert multiple SVG cards to PNG*

Description

Convert a list of SVG strings to PNG files in a given directory. Fonts are embedded automatically for consistent rendering.

Usage

```
batch_svg_to_png(
  svg_list,
  output_dir = ".",
  prefix = "card",
  dpi = 300,
  background = "transparent"
)
```

Arguments

`svg_list`	List of SVG strings.
`output_dir`	Output directory.
`prefix`	File name prefix.
`dpi`	Resolution.
`background`	Background color for PNG output.

Value

Character vector of output paths.

card_to_grob *Create a grob for grid/ggplot2*

Description

Wrap an SVG card as a raster grob so it can be used with grid graphics.

Usage

```
card_to_grob(svg_string, dpi = 150, engine = c("auto", "chrome", "rsvg"))
```

Arguments

`svg_string`	SVG string from `svg_card()`.
`dpi`	Rasterization DPI (default 150).
`engine`	Rendering engine: `"auto"`, `"chrome"`, or `"rsvg"`.

Value

A `grid::rasterGrob` object.

`chrome_available` *Check if Chrome/Chromium is available for rendering*

Description

Checks whether the chromote package can find and use a Chrome or Chromium installation for headless rendering.

Usage

```
chrome_available(verbose = FALSE)
```

Arguments

`verbose`	Print status messages (default FALSE).

Value

TRUE if Chrome is available, FALSE otherwise.

Examples

```
if (chrome_available()) {
   cat("Using Chrome")
} else {
   cat("Using Magick")
}
```

`compress_number` *Compress number to abbreviated format*

Description

Compress number to abbreviated format

Usage

```
compress_number(x, digits = 1)
```

Arguments

`x`	Numeric value to compress
`digits`	Number of decimal places

Value

Character string with abbreviated number

Examples

```
compress_number(1234567)
compress_number(36400000)
```

`create_badge` *Create an SVG Badge*

Description

Generate a badge similar to shields.io style with label and value. Both sides have properly rounded corners.

Usage

```
create_badge(
  label,
  value,
  color,
  font = "Jost",
  style = "margin:2px;",
  fontsize = 11,
  horiz_padding = 5,
  extra_right_pad = 2,
  class = "",
  shadow_offset = 2,
  corner_radius = 3,
  height = NULL,
  as_string = TRUE
)
```

Arguments

`label`	Label text (left side)
`value`	Value text (right side)
`color`	Background color for value area
`font`	Font family name
`style`	CSS style string
`fontsize`	Font size in pixels
`horiz_padding`	Horizontal padding
`extra_right_pad`	Extra padding on right side

`class`	CSS class
`shadow_offset`	Shadow offset in pixels
`corner_radius`	Corner radius for rounded rectangle
`height`	Minimum height (optional)
`as_string`	Return as character string

Value

SVG string

Examples

```
create_badge("UH", "192", "white")
create_badge("Recurso Federal", "36,4 milhões", "#4CAF50")
```

`create_badge_row` *Create a row of SVG badges with uniform height*

Description

Generate multiple badges arranged horizontally with the same height.

Usage

```
create_badge_row(
  badges_data,
  default_color = "white",
  spacing = 4,
  font = "Jost",
  fontsize = 10,
  uniform_height = TRUE
)
```

Arguments

`badges_data`	A list of lists, each containing label, value, and optionally color
`default_color`	Default color for badges
`spacing`	Spacing between badges
`font`	Font family
`fontsize`	Font size
`uniform_height`	Force all badges to have the same height (default TRUE)

Value

SVG string containing all badges

Examples

```
badges <- list(
  list(label = "UH", value = "192"),
  list(label = "Recurso Federal", value = "36,4 milhões"),
  list(label = "Contrapartida", value = "0,0")
)
create_badge_row(badges, default_color = "white")
```

---

create_bottom_logo_row

Create logo row for bottom-left corner of card

Description

Takes a list of SVG logos and arranges them horizontally for the bottom-left.

Usage

```
create_bottom_logo_row(
  logos,
  target_height = 30,
  spacing = 10,
  x_offset = 20,
  card_height = 400,
  y_offset = 20
)
```

Arguments

`logos`	List of SVG strings or file paths
`target_height`	Height for all logos (default 30)
`spacing`	Horizontal spacing between logos (default 10)
`x_offset`	Left margin from card edge
`card_height`	Total card height for y positioning
`y_offset`	Bottom margin

Value

A list with svg_content and total_width

`create_logo_row` *Create logo row for top-right corner of card*

Description

Takes a list of SVG logos and arranges them horizontally with proper spacing. Returns the SVG elements positioned for the top-right corner.

Usage

```
create_logo_row(
  logos,
  target_height = 40,
  spacing = 10,
  card_width = 500,
  x_offset = 20,
  y_offset = 20
)
```

Arguments

`logos`	List of SVG strings or file paths
`target_height`	Height for all logos (default 40)
`spacing`	Horizontal spacing between logos (default 10)
`card_width`	Total card width to calculate positioning
`x_offset`	Right margin from card edge
`y_offset`	Top margin

Value

A list with svg_content and total_width

`ensure_chrome` *Ensure Chrome is available, downloading if necessary*

Description

Checks if Chrome is available and optionally downloads a standalone Chrome for Testing if not found. This ensures Chrome-based rendering works without requiring a system-wide Chrome installation.

Usage

```
ensure_chrome(download = FALSE, verbose = TRUE)
```

Arguments

download	If TRUE and Chrome is not found, attempt to download Chrome for Testing (default FALSE).
verbose	Print status messages (default TRUE).

Details

When download = TRUE, this function will download "Chrome for Testing", a standalone Chrome distribution designed for automation. The download is approximately 150MB and is cached in the user's data directory.

Alternatively, you can:

- Install Chrome/Chromium system-wide
- Set the CHROMOTE_CHROME environment variable to point to an existing installation

Value

TRUE if Chrome is available (or was successfully downloaded), FALSE otherwise.

Examples

```
# Check and report status
ensure_chrome()

# Download Chrome if not available
## Not run:  # Requires an external Chrome/Chromium installation
ensure_chrome(download = TRUE)

## End(Not run)
```

find_chrome_path *Find Chrome executable path*

Description

Attempts to find a Chrome or Chromium executable on the system. Checks common installation paths and environment variables.

Usage

```
find_chrome_path()
```

Value

Path to Chrome executable, or NULL if not found.

Examples

```
path <- find_chrome_path()
if (!is.null(path)) {
  message("Chrome found at: ", path)
}
```

`font_available` *Check if a font is available for embedding*

Description

Check if a font is available for embedding

Usage

```
font_available(family)
```

Arguments

`family`	Font family name

Value

TRUE if font is cached (any supported format), FALSE otherwise

Examples

```
font_available("Jost")
font_available("Montserrat")
```

`font_cache_dir` *Get font cache directory*

Description

Returns the directory where cardargus caches downloaded font files. Fonts in this directory are automatically embedded in SVG/PNG exports.

Usage

```
font_cache_dir(persistent = TRUE)
```

Arguments

`persistent`	Logical. If TRUE (default), uses persistent cache via `tools::R_user_dir()`. If FALSE or if persistent cache is unavailable, uses a session-specific temporary directory.

Value

A character path to the cache directory.

`get_font_css` *Get Google Font CSS for embedding in SVG*

Description

Get Google Font CSS for embedding in SVG

Usage

```
get_font_css(font_name = "Jost", weights = c(400, 500, 600, 700))
```

Arguments

`font_name`	Name of the Google Font
`weights`	Vector of font weights to include

Value

Character string with CSS @font-face rules

Examples

```
get_font_css("Jost")
```

`get_svg_path` *Get path to a bundled SVG file*

Description

Returns the full path to a SVG file bundled with the package.

Usage

```
get_svg_path(filename, height = NULL, width = NULL)
```

Arguments

`filename`	Name of the SVG file (e.g., "morar_bem.svg")
`height`	Optional target height (px). If provided, returns the SVG content resized for embedding instead of the file path.
`width`	Optional target width (px). Only used when returning resized SVG content.

Value

If `height` and `width` are both NULL, returns the full file path. Otherwise returns the resized SVG content (character string).

`icon_building` *Building Icon SVG*

Description

Generate a building/apartment icon SVG.

Usage

```
icon_building(
  width = 50,
  height = 56,
  stroke_color = "white",
  stroke_width = 2,
  fill = "none"
)
```

Arguments

`width`	Width of the icon
`height`	Height of the icon
`stroke_color`	Stroke color
`stroke_width`	Stroke width
`fill`	Fill color

Value

SVG string

Examples

```
icon_building(50, 56)
```

icon_construction *Construction Icon SVG*

Description

Generate a construction/crane icon SVG.

Usage

```
icon_construction(
  width = 50,
  height = 56,
  stroke_color = "white",
  stroke_width = 2,
  fill = "none"
)
```

Arguments

`width`	Width of the icon
`height`	Height of the icon
`stroke_color`	Stroke color
`stroke_width`	Stroke width
`fill`	Fill color

Value

SVG string

Examples

```
icon_construction(50, 56)
```

icon_house *House Icon SVG*

Description

Generate a house/home icon SVG. You can also use any SVG file path instead of built-in icons.

Usage

```
icon_house(
  width = 50,
  height = 56,
  stroke_color = "white",
  stroke_width = 35,
  fill = "none"
)
```

Arguments

`width`	Width of the icon
`height`	Height of the icon
`stroke_color`	Stroke color
`stroke_width`	Stroke width
`fill`	Fill color (default none)

Value

SVG string

Examples

```
icon_house(50, 56)

# You can also use a custom SVG file:
# svg_card(..., with_icon = "/path/to/my_icon.svg")
```

`icon_map_pin` *Map Pin Icon SVG*

Description

Generate a map pin/location icon SVG.

Usage

```
icon_map_pin(
  width = 50,
  height = 56,
  stroke_color = "white",
  stroke_width = 2,
  fill = "none"
)
```

Arguments

`width`	Width of the icon
`height`	Height of the icon
`stroke_color`	Stroke color
`stroke_width`	Stroke width
`fill`	Fill color

Value

SVG string

Examples

```
icon_map_pin(50, 56)
```

`icon_money` *Dollar/Money Icon SVG*

Description

Generate a dollar/money icon SVG.

Usage

```
icon_money(
  width = 50,
  height = 56,
  stroke_color = "white",
  stroke_width = 2,
  fill = "none"
)
```

Arguments

`width`	Width of the icon
`height`	Height of the icon
`stroke_color`	Stroke color
`stroke_width`	Stroke width
`fill`	Fill color

Value

SVG string

Examples

```
icon_money(50, 56)
```

include_card *Display card in knitr/Quarto document (SVG via data URI)*

Description

Embeds the SVG as an `<img>` using a `data:image/svg+xml;base64,...` URI. This is more robust than inline <svg> for Pandoc/pkgdown (avoids "unclosed div" warnings). For non-HTML outputs, it falls back to `include_card_png()`.

Usage

```
include_card(
  svg_string,
  width = "100%",
  alt = "Card generated by cardargus",
  dpi = 300,
  engine = c("auto", "chrome", "rsvg")
)
```

Arguments

`svg_string`	SVG string from `svg_card()`.
`width`	Display width (CSS units), e.g. `"100%"`, `"500px"`.
`alt`	Alternative text for accessibility.
`dpi`	Fallback DPI used when output is not HTML (default 300).
`engine`	Rendering engine for non-HTML output: `"auto"`, `"chrome"`, or `"rsvg"`.

Value

Knitr output for the current format.

include_card_png *Display card as PNG in knitr/Quarto document*

Description

Converts an SVG card to PNG and displays it in R Markdown or Quarto documents. Recommended for better compatibility across HTML/PDF/Word.

Usage

```
include_card_png(
  svg_string,
  dpi = 300,
  width = "100%",
  alt = "Card generated by cardargus",
  background = "transparent",
  engine = c("auto", "chrome", "rsvg")
)
```

Arguments

`svg_string`	SVG string from `svg_card()`.
`dpi`	Rasterization DPI (default 300).
`width`	Display width (CSS units) for HTML outputs.
`alt`	Alternative text for accessibility.
`background`	Background color passed to `svg_to_png()` (default transparent).
`engine`	Rendering engine: `"auto"` (uses Chrome if available, else rsvg), `"chrome"` (headless Chrome via chromote), or `"rsvg"` (librsvg/magick). Chrome provides better font rendering for Google Fonts.

Value

Knitr output for the current format.

`install_fonts` *Pre-download fonts for offline use*

Description

Downloads and caches the specified fonts (or default fonts) so they are available for PNG conversion without internet access.

Usage

```
install_fonts(
  fonts = c("Jost", "Montserrat", "Roboto", "Open Sans"),
  verbose = TRUE
)
```

Arguments

`fonts`	Character vector of Google Font names to download. Default is c("Jost", "Montserrat", "Roboto", "Open Sans").
`verbose`	Print status messages

Value

Named logical vector indicating success for each font

Examples

```
install_fonts()
install_fonts(c("Jost", "Roboto"))
```

`is_light_color` *Check if a color is light*

Description

Check if a color is light

Usage

```
is_light_color(color)
```

Arguments

`color`	A hex color string or color name

Value

Logical indicating if the color is light

Examples

```
is_light_color("#FFFFFF")
is_light_color("#000000")
```

`list_bundled_svgs` *List available bundled SVG files*

Description

List available bundled SVG files

Usage

```
list_bundled_svgs()
```

Value

Character vector of available SVG filenames

`list_fonts` *List registered/cached fonts*

Description

List registered/cached fonts

Usage

```
list_fonts()
```

Value

Character vector of font family names that are cached

Examples

```
list_fonts()
```

`load_svg_for_embed` *Load and process external SVG file for embedding*

Description

Reads an SVG file and processes it to be embedded inside another SVG. Removes XML declarations, adjusts dimensions, and prepares for embedding.

Usage

```
load_svg_for_embed(svg_path, target_height = 40, target_width = NULL)
```

Arguments

`svg_path`	Path to the SVG file
`target_height`	Desired height in pixels
`target_width`	Optional desired width (calculated from aspect ratio if NULL)

Details

This function is useful when you want to embed custom logos or icons in cards. You can pass any SVG file path to the `logos`, `bottom_logos`, or `with_icon` parameters of `svg_card()`.

Value

A list with svg_content, width, and height

Examples

```
## Not run:  # Need a external svg file
# Load a custom logo
logo <- load_svg_for_embed("/path/to/logo.svg", target_height = 40)

# Or just pass the path directly to svg_card():
svg_card(
  title = "My Card",
  logos = c("/path/to/logo1.svg", "/path/to/logo2.svg"),
  ...
)

## End(Not run)
```

register_cardargus_knitr

Register cardargus knitr engine

Description

Registers a custom knitr engine named `"cardargus"` so that SVG cards can be rendered directly from chunks.

Usage

```
register_cardargus_knitr()
```

Value

Invisible NULL.

Examples

```
# In your setup chunk:
register_cardargus_knitr()

# Then use cardargus as chunk engine:
# ```{cardargus}
# svg_card(title = "My Card", ...)
# ```
```

`register_font` *Register a local font file for embedding*

Description

Copies a local font file (TTF or WOFF2) to the cardargus cache directory so it can be embedded in SVG exports.

Usage

```
register_font(font_path, family = NULL)
```

Arguments

`font_path`	Path to a local .ttf or .woff2 font file
`family`	Font family name to register (e.g., "Jost"). If NULL, the filename without extension is used.

Value

Path to the cached font file (invisible)

`register_google_font` *Register Google Font (sysfonts)*

Description

Registers a Google Font using **sysfonts**.

Usage

```
register_google_font(font_family)
```

Arguments

`font_family`	Font family name (e.g., "Jost")

Value

Invisible NULL

`save_card_for_knitr` *Save card and return path for knitr*

Description

Saves a card to a file and returns the path for use in knitr chunks.

Usage

```
save_card_for_knitr(
  svg_string,
  filename = "card",
  format = c("svg", "png"),
  dpi = 300,
  dir = NULL,
  engine = c("auto", "chrome", "rsvg")
)
```

Arguments

`svg_string`	SVG string from `svg_card()`.
`filename`	Output filename (without extension).
`format`	Output format: `"svg"` or `"png"`.
`dpi`	Rasterization DPI for PNG (default 300).
`dir`	Output directory (defaults to knitr figure directory or tempdir()).
`engine`	Rendering engine for PNG: `"auto"`, `"chrome"`, or `"rsvg"`.

Value

Path to the saved file.

`save_svg` *Save SVG string to file (sanitized + embedded fonts)*

Description

Saves an SVG string to disk. Before saving, the function:

1. sanitizes the SVG to remove problematic Inkscape/Sodipodi metadata that can break strict XML parsers, and
2. detects and embeds fonts (WOFF2 via @font-face) for deterministic rendering.

This function expects the font helpers to be available in the package: `detect_svg_fonts()`, `ensure_cardargus_fonts()`, and `embed_svg_fonts()`.

Usage

```
save_svg(svg_content, output_path)
```

Arguments

`svg_content`	SVG string (or object coercible to character).
`output_path`	Output file path.

Value

Path to the saved SVG file.

Examples

```
svg <- svg_card("FAR", list(), list())
save_svg(svg, tempfile(fileext = ".svg"))
```

`setup_fonts` *Setup showtext for cardargus*

Description

Registers fonts (via sysfonts) and optionally enables showtext auto mode.

Usage

```
setup_fonts(fonts = c("Jost", "Montserrat"), auto = TRUE)
```

Arguments

`fonts`	Character vector of Google Font names to register.
`auto`	Enable showtext auto mode.

Value

Invisible NULL

svgs_dir *Get the path to package SVGs directory*

Description

Returns the path to the inst/svgs directory where SVG files are stored.

Usage

```
svgs_dir()
```

Value

Character string with the path to SVGs directory

svg_card *Create an Information Card in SVG Format*

Description

Generate a complete information card as an SVG with embedded styles, fonts, badges, logos, and field labels/values.

Usage

```
svg_card(
  title = "FAR",
  badges_data = list(),
  fields = list(),
  bg_color = "#fab255",
  width = 500,
  padding = 20,
  corner_radius = 8,
  font = "Jost",
  title_fontsize = 16,
  title_color = "white",
  label_fontsize = 11,
  value_fontsize = 11,
  label_color = "white",
  value_bg_color = "#f8f8ff",
  value_text_color = "#212529",
  show_house_icon = TRUE,
  logos = list(),
  logos_height = 40,
  bottom_logos = list(),
  bottom_logos_height = 30,
  footer = NULL,
  gap_to_footer = 6,
  footer_row_padding_bottom = 6,
  footer_fontsize = 8,
  footer_color = "white",
  uniform_row_height = TRUE,
  show_viewer = interactive()
)
```

Arguments

`title`	Card title (e.g., "FAR", "FNHIS")
`badges_data`	List of badge data (label, value, color)
`fields`	List of field rows, each row is a list of fields with label, value, and optionally with_icon. The with_icon parameter can be: • TRUE - uses the default house icon • FALSE or NULL - no icon • A character string - path to an SVG file or raw SVG code
`bg_color`	Background color of the card. Can be a solid color (e.g., "#fab255") or a CSS gradient (e.g., "linear-gradient(to right, #1a5a3a, #2e7d32)" or "linear-gradient(135deg, #667eea, #764ba2)").
`width`	Card width in pixels
`padding`	Padding inside the card
`corner_radius`	Corner radius for rounded corners
`font`	Font family
`title_fontsize`	Title font size
`title_color`	Color for the card title (default "white")
`label_fontsize`	Label font size
`value_fontsize`	Value font size
`label_color`	Color for field labels (default "white")
`value_bg_color`	Background color for value boxes
`value_text_color`	Text color for values
`show_house_icon`	Show house icon next to empreendimento
`logos`	Character vector of logo file paths or SVG strings for top right. Use `get_svg_path("filename.svg")` for bundled logos, or any local path.
`logos_height`	Height for top-right logos (default 40)
`bottom_logos`	Character vector of logo file paths or SVG strings for bottom left.
`bottom_logos_height`	Height for bottom-left logos (default 30)

`footer`	Footer text (e.g., update timestamp)
`gap_to_footer`	Distance (px) between the last info block and the footer row.
`footer_row_padding_bottom`	Bottom padding (px) under the footer row (text + logos).
`footer_fontsize`	Footer font size
`footer_color`	Color for footer text (default "white")
`uniform_row_height`	If TRUE, keep the height inside a row.
`show_viewer`	If TRUE (and interactive), preview the SVG in the Viewer.

Value

SVG string

Examples

```
# With default house icon
fields <- list(
  list(
    list(label = "Empreendimento", value = "CAIARA II", with_icon = TRUE)
  )
)

# With custom icon
custom_icon <- '<svg width="50" height="50"><circle cx="25" cy="25" r="20" fill="white"/></svg>'
fields <- list(
  list(
    list(label = "Projeto", value = "Meu Projeto", with_icon = custom_icon)
  )
)

badges <- list(
  list(label = "UH", value = "192"),
  list(label = "Recurso Federal", value = "36,4 milhões")
)

# With file paths for logos
svg_card("FAR", badges, fields,
         bg_color = "#fab255",
         logos = c("path/to/logo1.svg", "path/to/logo2.svg"),
         bottom_logos = c("path/to/gov_logo.svg"))

# With gradient background
svg_card("MCMV", badges, fields,
         bg_color = "linear-gradient(to right, #1a5a3a, #2e7d32)")

# Diagonal gradient
svg_card("Programa", badges, fields,
         bg_color = "linear-gradient(135deg, #667eea, #764ba2)")
```

`svg_to_formats` *Convert SVG to multiple formats*

Description

Convert an SVG string or file to multiple formats. Supported formats are:

- `"svg"` - saves the SVG
- `"png"` - rasterizes to PNG via `svg_to_png`
- `"pdf"` - converts to PDF (prefers **rsvg**)

Usage

```
svg_to_formats(
  svg_input,
  output_base,
  formats = c("svg", "png"),
  dpi = 300,
  background = "transparent"
)
```

Arguments

`svg_input`	SVG string or file path.
`output_base`	Base name for output files (without extension).
`formats`	Vector of formats to generate ("png", "svg", "pdf").
`dpi`	Resolution for raster formats.
`background`	Background color for PNG output.

Value

Named list with paths to generated files.

`svg_to_pdf_chrome` *Convert SVG to PDF using headless Chrome*

Description

Renders an SVG to PDF using headless Chrome via the chromote package. This method produces vector PDFs with perfect font rendering.

Usage

```
svg_to_pdf_chrome(
  svg_input,
  output_path,
  background = "transparent",
  print_background = TRUE,
  timeout = 30
)
```

Arguments

`svg_input`	SVG string or path to an SVG file.
`output_path`	Output path for the PDF file.
`background`	Background color for the HTML page (default "transparent").
`print_background`	Whether to include CSS backgrounds in PDF (default TRUE).
`timeout`	Maximum time in seconds to wait for page load (default 30).

Value

Path to the generated PDF file.

Examples

```
## Not run:  # It requires an external Chrome/Chromium installation
svg <- svg_card("FAR", list(), list())
if (chrome_available()) {
  pdf_path <- svg_to_pdf_chrome(svg, tempfile(fileext = ".pdf"))
}

## End(Not run)
```

`svg_to_png` *Convert SVG to PNG*

Description

Convert an SVG string or SVG file path to a high-quality PNG image. The function sanitizes the SVG and embeds required WOFF2 fonts (downloaded on demand into a user cache) to ensure consistent font rendering.

Important note about DPI: `rsvg` rasterizes primarily based on pixel dimensions. To make DPI matter, this function scales output pixel size by `(dpi / 96)` when `width` / `height` are not explicitly provided.

Usage

```
svg_to_png(
  svg_input,
  output_path = NULL,
  width = NULL,
  height = NULL,
  dpi = 300,
  background = "transparent"
)
```

Arguments

`svg_input`	SVG string or path to an SVG file.
`output_path`	Output path for the PNG file (optional; a temp file is used if NULL).
`width`	Output width in pixels (NULL to infer from SVG and scale by DPI).
`height`	Output height in pixels (NULL to infer from SVG and scale by DPI).
`dpi`	Resolution in dots per inch (default 300 for high quality).
`background`	Background color. Use "transparent" or "none" for transparency (default), or specify a color like "white", "#FFFFFF", etc.

Value

Path to the generated PNG file.

Examples

```
svg <- svg_card("FAR", list(), list())
file_name <- tempfile(fileext = ".png")
png_path <- svg_to_png(svg, file_name, dpi = 300)
png_path <- svg_to_png(svg, file_name, dpi = 300, background = "white")
```

`svg_to_png_chrome` *Convert SVG to PNG using headless Chrome*

Description

Renders an SVG to PNG using headless Chrome via the chromote package. This method provides superior font rendering compared to librsvg/ImageMagick, as Chrome properly handles @font-face rules, web fonts, and CSS features.

Usage

```
svg_to_png_chrome(
  svg_input,
  output_path = NULL,
  dpi = 300,
  background = "transparent",
  timeout = 30
)
```

Arguments

`svg_input`	SVG string or path to an SVG file.
`output_path`	Output path for the PNG file. If NULL, a temp file is used.
`dpi`	Resolution in dots per inch (default 300). Chrome uses 96 DPI as base, so dpi = 300 results in approximately 3.125x scaling.
`background`	Background color for the HTML page (default "transparent"). Use "white", "#FFFFFF", etc. for a solid background.
`timeout`	Maximum time in seconds to wait for page load (default 30).

Value

Path to the generated PNG file.

Examples

```
svg <- svg_card("FAR", list(), list())
file_name <- tempfile(fileext = ".png")
# High-quality PNG with Chrome rendering
## Not run:  # Requires an external Chrome/Chromium installation.
if (chrome_available()) {
  png_path <- svg_to_png_chrome(svg, file_name, dpi = 300)
}

## End(Not run)
```

`svg_to_png_with_margin` *Convert SVG to PNG with optional margin and background*

Description

Creates a PNG with extra margin around the card. Fonts are embedded before rasterization for consistent appearance.

Usage

```
svg_to_png_with_margin(
  svg_input,
  output_path = NULL,
  margin = 20,
  margin_color = "transparent",
  dpi = 300,
  background = "transparent"
)
```

Arguments

`svg_input`	SVG string or path to SVG file.
`output_path`	Output path for PNG file (optional; a temp file is used if NULL).
`margin`	Margin in pixels to add around the card.
`margin_color`	Color of the margin area (default transparent).
`dpi`	Resolution in dots per inch.
`background`	Background color for the card rasterization (default transparent).

Value

Path to the generated PNG file.

Index