Package 'string2path'

November 22, 2021

Title Rendering Font into 'data.frame'

Version 0.0.4

Description Extract glyph information from a font file, and translate the outline curves to flattened paths or tessellated polygons. The converted data is returned as a 'data.frame' in easy-to-plot format.

License MIT + file LICENSE

Depends R (>= 4.0)

Imports tibble

Suggests testthat (>= 3.0.0), systemfonts

URL https://yutannihilation.github.io/string2path/, https://github.com/yutannihilation/string2path

BugReports https://github.com/yutannihilation/string2path/issues

Encoding UTF-8

RoxygenNote 7.1.2

SystemRequirements Cargo (rustc package manager)

Biarch true

Config/testthat/edition 3

Config/string2path/MSRV 1.41.0

Config/string2path/windows_toolchain stable-msvc

Config/string2path/crate_name string2path

Config/string2path/github_repo yutannihilation/string2path

Config/string2path/github_tag build_20210921-1

Config/string2path/binary_sha256sum list(
`aarch64-apple-darwin-libstring2path.a` =
``4a34f99cec66610746b20d456b1e11b346596c22ea1935c1bcb5ef1ab725f0e8",
`i686-pc-windows-gnu-libstring2path.a` =
``ceda54184fb3bf9e4cbba86848cb2091ff5b77870357f94319f9215fadfa5b25",
`ucrt-x86_64-pc-windows-gnu-libstring2path.a` =
``26a05f6ee8c2f625027ffc77c97fc8ac9746a182f5bc53d64235999a02c0b0dc",

`x86_64-apple-darwin-libstring2path.a` =
``be65f074cb7ae50e5784e7650f48579fff35f30ff663d1c01eabdc9f35c1f87c",
`x86_64-pc-windows-gnu-libstring2path.a` =
``26a05f6ee8c2f625027ffc77c97fc8ac9746a182f5bc53d64235999a02c0b0dc"
)

NeedsCompilation yes

Author Hiroaki Yutani [aut, cre] (<https://orcid.org/0000-0002-3385-7233>)

Maintainer Hiroaki Yutani <yutani.ini@gmail.com>

Repository CRAN

Date/Publication 2021-11-22 12:50:02 UTC

R topics documented:

`string2path` *Convert a String to Paths*

Description

`string2path()` converts a text to the paths of the width-less outlines of each glyph. `string2stroke()` converts a text to the paths of the outlines, with the specified line width, of each glyph. `string2fill()` converts a text to the paths of the filled polygon of each glyph.

Usage

```
string2path(text, font_file, tolerance = 5e-05)

string2stroke(text, font_file, tolerance = 5e-05, line_width = 0.03)

string2fill(text, font_file, tolerance = 5e-05)
```

Arguments

`text`	A text to convert to paths.
`font_file`	A path to a 'TrueType' font file ('.ttf') or an 'OpenType' font file ('.otf').
`tolerance`	Maximum distance allowed between the curve and its approximation. For more details, please refer to the documentation of the underlying Rust library.
`line_width`	Line width of strokes.

Value

A `tibble()` containing these columns:

x x position of the point on the path, scaled to x / line height. The left side of the first glyph is at x = 0.

y Y position of the point on the path, scaled to y / line height. The baseline of the first line is at y = 0.

glyph_id IDs to distinguish the glyphs.

path_id IDs to distinguish the groups of paths.

triangle_id IDs to distinguish the triangles. `string2path()` doesn't contain this column.

Examples

```
if (requireNamespace("systemfonts", quietly = TRUE)) {
  available_fonts <- systemfonts::system_fonts()$path

  # string2path supports only TrueType or OpenType formats
  ttf_or_otf <- available_fonts[grepl("\\.(ttf|otf)$", available_fonts)]

  # string2path() converts a text to paths
  d_path <- string2path("TEXT", ttf_or_otf[1])
  plot(d_path$x, d_path$y)
  for (p in split(d_path, d_path$path_id)) {
    lines(p$x, p$y)
  }

  # string2stroke() converts a text to strokes
  d_stroke <- string2stroke("TEXT", ttf_or_otf[1])
  plot(d_stroke$x, d_stroke$y)

  # The stroke is split into triangles, which can be distinguished by `triangle_id`
  set.seed(2)
  for (p in split(d_stroke, d_stroke$triangle_id)) {
    polygon(p$x, p$y, col = rgb(runif(1), runif(1), runif(1), 0.8))
  }

  # string2fill() converts a text to filled polygons
  d_fill <- string2fill("TEXT", ttf_or_otf[1])
  plot(d_fill$x, d_fill$y)

  # The polygon is split into triangles, which can be distinguished by `triangle_id`
  set.seed(2)
  for (p in split(d_fill, d_fill$triangle_id)) {
    polygon(p$x, p$y, col = rgb(runif(1), runif(1), runif(1), 0.8))
  }
}
```

Index