Package: textshaping (via r-universe)

September 24, 2024

Title Bindings to the 'HarfBuzz' and 'Fribidi' Libraries for Text Shaping

Version 0.4.0.9000

Description Provides access to the text shaping functionality in the 'HarfBuzz' library and the bidirectional algorithm in the 'Fribidi' library. 'textshaping' is a low-level utility package mainly for graphic devices that expands upon the font tool-set provided by the 'systemfonts' package.

License MIT + file LICENSE

URL https://github.com/r-lib/textshaping

BugReports https://github.com/r-lib/textshaping/issues

Depends R (>= 3.2.0)

Imports lifecycle, systemfonts (>= 1.1.0)

Suggests covr, knitr, rmarkdown

LinkingTo cpp11 (>= 0.2.1), systemfonts (>= 1.0.0)

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

SystemRequirements freetype2, harfbuzz, fribidi

Repository https://r-lib.r-universe.dev

RemoteUrl https://github.com/r-lib/textshaping

RemoteRef HEAD

RemoteSha 0cf6a497a439ad400d6c7f503072b3558a86c01b

Contents

`get_font_features` *Get available OpenType features in a font*

Description

This is a simply functions that returns the available OpenType feature tags for one or more fonts. See `font_feature()` for more information on how to use the different feature with a font.

Usage

```
get_font_features(
  family = "",
  italic = FALSE,
  bold = FALSE,
  path = NULL,
  index = 0
)
```

Arguments

`family`	The name of the font families to match
`italic`	logical indicating the font slant
`bold`	logical indicating whether the font weight
`path, index`	path an index of a font file to circumvent lookup based on family and style

Value

A list with an element for each of the input fonts containing the supported feature tags for that font.

Examples

```
# Select a random font on the system
sys_fonts <- systemfonts::system_fonts()
random_font <- sys_fonts$family[sample(nrow(sys_fonts), 1)]

# Get the features
get_font_features(random_font)
```

shape_text *Calculate glyph positions for strings*

Description

[Experimental]

Do basic text shaping of strings. This function will use freetype to calculate advances, doing kerning if possible. It will not perform any font substitution or ligature resolving and will thus be much in line with how the standard graphic devices does text shaping. Inputs are recycled to the length of `strings`.

Usage

```
shape_text(
  strings,
  id = NULL,
  family = "",
  italic = FALSE,
  weight = "normal",
  width = "normal",
  features = font_feature(),
  size = 12,
  res = 72,
  lineheight = 1,
  align = "left",
  hjust = 0,
  vjust = 0,
  max_width = NA,
  tracking = 0,
  indent = 0,
  hanging = 0,
  space_before = 0,
  space_after = 0,
  path = NULL,
  index = 0,
  bold = deprecated()
)
```

Arguments

`strings`	A character vector of strings to shape
`id`	A vector grouping the strings together. If strings share an id the shaping will continue between strings
`family`	The name of the font families to match
`italic`	logical indicating the font slant

weight	The weight to query for, either in numbers (0, 100, 200, 300, 400, 500, 600, 700, 800, or 900) or strings ("undefined", "thin", "ultralight", "light", "normal", "medium", "semibold", "bold", "ultrabold", or "heavy"). NA will be interpreted as "undefined"/0
width	The width to query for either in numbers (0, 1, 2, 3, 4, 5, 6, 7, 8, or 9) or strings ("undefined", "ultracondensed", "extracondensed", "condensed", "semicondensed", "normal", "semiexpanded", "expanded", "extraexpanded", or "ultraexpanded"). NA will be interpreted as "undefined"/0
features	A systemfonts::font_feature() object or a list of them, giving the OpenType font features to set
size	The size in points to use for the font
res	The resolution to use when doing the shaping. Should optimally match the resolution used when rendering the glyphs.
lineheight	A multiplier for the lineheight
align	Within text box alignment, either 'left', 'center', 'right', 'justified-left', 'justified-right', 'justified-center', or 'distributed'
hjust, vjust	The justification of the textbox surrounding the text
max_width	The requested with of the string in inches. Setting this to something other than NA will turn on word wrapping.
tracking	Tracking of the glyphs (space adjustment) measured in 1/1000 em.
indent	The indent of the first line in a paragraph measured in inches.
hanging	The indent of the remaining lines in a paragraph measured in inches.
space_before, space_after	The spacing above and below a paragraph, measured in points
path, index	path an index of a font file to circumvent lookup based on family and style
bold	logical indicating whether the font weight

Value

A list with two element: shape contains the position of each glyph, relative to the origin in the enclosing textbox. metrics contain metrics about the full strings.

shape is a data.frame with the following columns:

glyph The glyph as a character

index The index of the glyph in the font file

metric_id The index of the string the glyph is part of (referencing a row in the metrics data.frame)

string_id The index of the string the glyph came from (referencing an element in the strings input)

x_offset The x offset in pixels from the origin of the textbox

y_offset The y offset in pixels from the origin of the textbox

x_mid The x offset in pixels to the middle of the glyph, measured from the origin of the glyph

metrics is a data.frame with the following columns:

string The text the string consist of

width The width of the string

height The height of the string

left_bearing The distance from the left edge of the textbox and the leftmost glyph

right_bearing The distance from the right edge of the textbox and the rightmost glyph

top_bearing The distance from the top edge of the textbox and the topmost glyph

bottom_bearing The distance from the bottom edge of the textbox and the bottommost glyph

left_border The position of the leftmost edge of the textbox related to the origin

top_border The position of the topmost edge of the textbox related to the origin

pen_x The horizontal position of the next glyph after the string

pen_y The vertical position of the next glyph after the string

Examples

```
string <- "This is a long string\nLook; It spans multiple lines\nand all"

# Shape with default settings
shape_text(string)

# Mix styles within the same string
string <- c(
  "This string will have\na ",
  "very large",
  " text style\nin the middle"
)

shape_text(string, id = c(1, 1, 1), size = c(12, 24, 12))
```

`text_width` *Calculate the width of a string, ignoring new-lines*

Description

This is a very simple alternative to `shape_string()` that simply calculates the width of strings without taking any newline into account. As such it is suitable to calculate the width of words or lines that has already been splitted by `\n`. Input is recycled to the length of `strings`.

Usage

```
text_width(
  strings,
  family = "",
  italic = FALSE,
  bold = FALSE,
```

```
  size = 12,
  res = 72,
  include_bearing = TRUE,
  path = NULL,
  index = 0
)
```

Arguments

`strings`	A character vector of strings
`family`	The name of the font families to match
`italic`	logical indicating the font slant
`bold`	logical indicating whether the font weight
`size`	The pointsize of the font to use for size related measures
`res`	The ppi of the size related mesures
`include_bearing`	Logical, should left and right bearing be included in the string width?
`path`, `index`	path an index of a font file to circumvent lookup based on family and style

Value

A numeric vector giving the width of the strings in pixels. Use the provided `res` value to convert it into absolute values.

Examples

```
strings <- c('A short string', 'A very very looong string')
text_width(strings)
```

Index