

# Package ‘textshaping’

May 8, 2026

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

**Version** 1.0.5

**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, stats, stringi, systemfonts (>= 1.3.0), utils

**Suggests** covr, grDevices, grid, knitr, rmarkdown, testthat (>= 3.0.0)

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

**VignetteBuilder** knitr

**Config/build/compilation-database** true

**Config/testthat/edition** 3

**Config/usethis/last-upkeep** 2025-04-23

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**SystemRequirements** freetype2, harfbuzz, fribidi

**NeedsCompilation** yes

**Author** Thomas Lin Pedersen [cre, aut] (ORCID: <https://orcid.org/0000-0002-5147-4711>),  
Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

**Maintainer** Thomas Lin Pedersen <thomas.pedersen@posit.co>

**Repository** CRAN

**Date/Publication** 2026-03-06 10:40:02 UTC

## Contents

get_font_features . . . . .	2
lorem_text . . . . .	3
plot_shape . . . . .	4
shape_text . . . . .	5
text_width . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

get_font_features	<i>Get available OpenType features in a font</i>
-------------------	--

---

### 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	<b>[Deprecated]</b> Use weight = "bold" instead
path, index	path and 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)
```

---

`lorem_text`*Get gibberish text in various scripts*

---

## Description

Textshaping exists partly to allow all the various scripts that exists in the world to be used in R graphics. This function returns gibberish filler text (lorem ipsum text) in various scripts for testing purpose. Some of these are transliterations of the original lorem ipsum text while others are based an a distribution model.

## Usage

```
lorem_text(  
  script = c("latin", "chinese", "arabic", "devanagari", "cyrillic", "kana", "hangul",  
            "greek", "hebrew", "armenian", "georgian"),  
  n = 1  
)  
  
lorem_bidi(  
  ltr = c("latin", "chinese", "devanagari", "cyrillic", "kana", "hangul", "greek",  
         "armenian", "georgian"),  
  rtl = c("arabic", "hebrew"),  
  ltr_prop = 0.9,  
  n = 1  
)
```

## Arguments

<code>script</code>	A string giving the script to fetch gibberish for
<code>n</code>	The number of paragraphs to fetch. Each paragraph will be its own element in the returned character vector.
<code>ltr, rtl</code>	scripts to use for left-to-right and right-to-left text
<code>ltr_prop</code>	The approximate proportion of left-to-right text in the final string

## Value

a character vector of length `n`

## References

<https://generator.lorem-ipsum.info>

## Examples

```
# Defaults to standard lorem ipsum
lorem_text()

# Get two paragraphs of hangul (Korean)
lorem_text("hangul", 2)

# Get gibberish bi-directional text
lorem_bidi()
```

---

plot\_shape

*Preview shaped text and the metrics for the text box*

---

## Description

This function allows you to preview the layout that `shape_text()` calculates. It is purely meant as a sanity check to make sure that the values calculated are sensible and shouldn't be used as a plotting function for rendering text on its own.

## Usage

```
plot_shape(shape, id = 1)
```

## Arguments

shape	The output of a call to <code>shape_text()</code>
id	The index of the text run to show in case shape contains multiples

## Value

This function is called for its side effects

## Examples

```
arab_text <- lorem_text("arabic", 2)
shape <- shape_text(
  arab_text,
  max_width = 5,
  indent = 0.2
)

try(
  plot_shape(shape)
)
```

---

shape_text	<i>Calculate glyph positions for strings</i>
------------	--

---

### Description

Performs advanced text shaping of strings including font fallbacks, bidirectional script support, word wrapping and various character and paragraph level formatting settings.

### Usage

```
shape_text(
  strings,
  id = NULL,
  family = "",
  italic = FALSE,
  weight = "normal",
  width = "undefined",
  features = font_feature(),
  size = 12,
  res = 72,
  lineheight = 1,
  align = "auto",
  hjust = 0,
  vjust = 0,
  max_width = NA,
  tracking = 0,
  indent = 0,
  hanging = 0,
  space_before = 0,
  space_after = 0,
  direction = "auto",
  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 <code>systemfonts::font_feature()</code> 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 'auto', 'left', 'center', 'right', 'justified', 'justified-left', 'justified-right', 'justified-center', or 'distributed'. 'auto' and 'justified' will chose the left or right version depending on the direction of the text.
hjust, vjust	The justification of the textbox surrounding the text
max_width	The requested width 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
direction	The overall directional flow of the text. The default ("auto") will guess the direction based on the content of the string. Use "ltr" (left-to-right) and "rtl" (right-to-left) to turn detection of and set it manually.
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 placement of the the first character contributing to the glyph within the string

**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

**font\_path** The path to the font file used during shaping of the glyph

**font\_index** The index of the font used to shape the glyph in the font file

**font\_size** The size of the font used during shaping

**advance** The advancement amount to the next glyph

**ascender** The ascend of the font used for the glyph. This does not measure the actual glyph

**descender** The descend of the font used for the glyph. This does not measure the actual 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

**ltr** The global direction of the string. If TRUE then it is left-to-right, otherwise it is right-to-left

## 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	<i>Calculate the width of a string, ignoring new-lines</i>
------------	--

---

### Description

This is a very simple alternative to `systemfonts::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 have already been split by `\n`. Input is recycled to the length of strings.

### Usage

```
text_width(
  strings,
  family = "",
  italic = FALSE,
  weight = "normal",
  width = "undefined",
  features = font_feature(),
  size = 12,
  res = 72,
  include_bearing = TRUE,
  path = NULL,
  index = 0,
  bold = deprecated()
)
```

### Arguments

strings	A character vector of 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 <code>systemfonts::font_feature()</code> 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.

<code>include_bearing</code>	Logical, should left and right bearing be included in the string width?
<code>path, index</code>	path an index of a font file to circumvent lookup based on family and style
<code>bold</code>	logical indicating whether the font weight

**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

`font_feature()`, 2

`get_font_features`, 2

`lorem_bidi (lorem_text)`, 3

`lorem_text`, 3

`plot_shape`, 4

`shape_text`, 5

`shape_text()`, 4

`systemfonts::font_feature()`, 6, 8

`systemfonts::shape_string()`, 8

`text_width`, 8