Package 'lofifonts'

November 8, 2024

Type Package

Title Text Rendering with Bitmap and Vector Fonts

Version 0.1.3

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description Alternate font rendering is useful when rendering text to novel graphics outputs where modern font rendering is not available or where bespoke text positioning is required. Bitmap and vector fonts allow for custom layout and rendering using pixel coordinates and line drawing. Formatted text is created as a data.frame of pixel coordinates (for bitmap fonts) or stroke coordinates (for vector fonts). All text can be easily previewed as a matrix or raster image.
A selection of fonts is included with this package.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/coolbutuseless/lofifonts

BugReports https://github.com/coolbutuseless/lofifonts/issues

Depends R (>= 2.10)

Suggests knitr, rmarkdown, bittermelon, ggplot2, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Mike Cheng [aut, cre, cph],
June Choe [ctb] (Contributed character positioning code),
Frederic Cambus [cph, tyd] (Creator of 'spleen' font),
GNU Unifont authors [cph, tyd] (Creators of 'unifont'),
Suraj Kurapati [cph, tyd] (Creator of 'Tamzen' font),
Scott Fial [cph, tyd] (Creator of 'Tamsyn' font upon which 'Tamzen' is based),
Anders Hoff [cph, tyd] (Creator of 'gridfont' font)

Repository CRAN

Date/Publication 2024-11-08 15:20:02 UTC

Contents

`bitmap_text_coords` *Create a data.frame of pixel coordinate information of the rendered text*

Description

Create a data.frame of pixel coordinate information of the rendered text

Usage

```
bitmap_text_coords(text, font = "unifont", dx = 0L, dy = 0L, missing = NULL)
```

Arguments

`text`	Single text string. Can include carriage returns to split text over multiple lines.
`font`	Name of bitmap font, or a 'lofi' font object. Default: 'unifont'. Use `get_lofi_names('bitmap')` to retrieve a list of all valid bitmap fonts included in this package. To create a 'lofi' font object use `convert_bm_font_to_lofi()`

dx	Additional character spacing in the horizontal direction. Default: 0
dy	Additional character spacing in the vertical direction i.e. between rows of text. Default: 0
missing	Codepoint to use if glyph not found in font. Default: NULL means to use the default specified by the font internally. Otherwise it will default to the codepoint for '?'

Value

data.frame of coordinate information

`char_idx` The index of the glyph within the provided `text` string

`codepoint` Unicode codepoint (integer)

`x` Pixel coordinate x value for display

`y` Pixel coordinate y value for display

`line` Line number within input `text` where this character appears

`x0` Original untransformed x-coordinate

`y0` Original untransformed y-coordinate

See Also

Other bitmap text functions: `bitmap_text_matrix()`, `bitmap_text_raster()`

Examples

```
bitmap_text_coords('Hi')
```

`bitmap_text_matrix` *Create a binary matrix of the rendered text*

Description

Create a binary matrix of the rendered text

Usage

```
bitmap_text_matrix(
  text,
  font = "unifont",
  dx = 0L,
  dy = 0L,
  scale_matrix = 1,
  missing = NULL
)
```

Arguments

text	Single text string. Can include carriage returns to split text over multiple lines.
font	Name of bitmap font, or a 'lofi' font object. Default: 'unifont'. Use `get_lofi_names('bitmap')` to retrieve a list of all valid bitmap fonts included in this package. To create a 'lofi' font object use `convert_bm_font_to_lofi()`
dx	Additional character spacing in the horizontal direction. Default: 0
dy	Additional character spacing in the vertical direction i.e. between rows of text. Default: 0
scale_matrix	Integer size scale factor. Default: 1. Must be an integer value >= 1. Scale up the matrix or raster result by this factor
missing	Codepoint to use if glyph not found in font. Default: NULL means to use the default specified by the font internally. Otherwise it will default to the codepoint for '?'

Value

Binary matrix representation of the rendered text

See Also

Other bitmap text functions: `bitmap_text_coords()`, `bitmap_text_raster()`

Examples

```
bitmap_text_matrix('Hi')
```

---

bitmap_text_raster *Create a raster image of the rendered text*

Description

Create a raster image of the rendered text

Usage

```
bitmap_text_raster(
  text,
  font = "unifont",
  dx = 0L,
  dy = 0L,
  scale_matrix = 1,
  missing = NULL
)
```

Arguments

text	Single text string. Can include carriage returns to split text over multiple lines.
font	Name of bitmap font, or a 'lofi' font object. Default: 'unifont'. Use `get_lofi_names('bitmap')` to retrieve a list of all valid bitmap fonts included in this package. To create a 'lofi' font object use `convert_bm_font_to_lofi()`
dx	Additional character spacing in the horizontal direction. Default: 0
dy	Additional character spacing in the vertical direction i.e. between rows of text. Default: 0
scale_matrix	Integer size scale factor. Default: 1. Must be an integer value >= 1. Scale up the matrix or raster result by this factor
missing	Codepoint to use if glyph not found in font. Default: NULL means to use the default specified by the font internally. Otherwise it will default to the codepoint for '?'

Value

Raster image representation of the rendered text

See Also

Other bitmap text functions: `bitmap_text_coords()`, `bitmap_text_matrix()`

Examples

```
ras <- bitmap_text_raster('Hi')
plot(ras)
```

---

convert_bm_font_to_lofi

Convert a 'bittermelon' 'bm_font' to a lofi font

Description

Convert a 'bittermelon' 'bm_font' to a lofi font

Usage

```
convert_bm_font_to_lofi(font, font_name = "Unknown")
```

Arguments

font	font object of class `bm_font`
font_name	Name of font

Value

lofi font object

Examples

```
filename <- system.file("fonts/spleen/spleen-8x16.hex.gz", package = "bittermelon")
bmfont <- bittermelon::read_hex(filename)
lofi <- convert_bm_font_to_lofi(bmfont)
lofi
```

---

get_lofi_font *Fetch an included 'lofi' font*

Description

Fetch an included 'lofi' font

Usage

```
get_lofi_font(font_name)
```

Arguments

font_name	Name of font e.g. 'unifont'

Value

'lofi' font object. See lofi for details of this data structure

Examples

```
get_lofi_font('unifont')
```

---

get_lofi_names *Return the names of all included fonts*

Description

Return the names of all included fonts

Usage

```
get_lofi_names(type)
```

Arguments

type	font type. Either 'bitmap', 'vector' or 'all'

Value

List of two elements: names of bitmap fonts, names of vector fonts

Examples

```
get_lofi_names('bitmap')
```

---

lofi *A description of the 'lofi' font format used to store fonts for this package*

Description

This package uses a custom data structure to store font information. This data structure is optimized for access to random sequences of glyphs which can be quickly assembled into a data.frame of points (for bitmap fonts) or strokes (for vector fonts). This data structure also needs to be compact and avoid unnecessary repetition. This is because the 'unifont' font contains pixel coordinates for over 100,000 codepoints and all this data must not exceed package size limitations for CRAN.

Details

Each 'lofi' font is a list object with the following members:

coords A data frame of 'x', 'y' coordinates. For vector fonts this also includes a 'stroke_idx' to delineate the individual strokes within a single glyph. This is a simple concatenation of *all* points (or strokes) in a font. Extracting this font data for a particular codepoint requires the use of other indexing elements in the 'lofi' structure. NOTE: For bitmap fonts, (x, y) coordinates must be numeric, integer or raw values with no values below zero.

codepoint_to_idx An integer vector. Use a codepoint (integer) to access the row index into the 'glyph_info' data.frame which holds the meta-information about this glyph. Because codepoints are indexed from 0, but R indexes vectors from 1, to access the row index: `codepoint_to_idx[codepoint + 1]`

line_height Integer. The line height of this font in pixels

default_codepoing Integer. The default unicode codepoint to use if the font does not contain a given glyph

baseline_offset Numeric value. The offset between the bottom of the font data and the baseline for the text

name name of font

glyph_info A data.frame of meta-information about each glyph. One row per glyph

codepoint Glyph codepoint (integer value)

npoints The number of rows of data in 'coords' data.frame which are used to define this font

row_start The index of the first row in 'coords' data.frame which contains data for this font

row_end The index of the last row in 'coords' data.frame which contains data for this font

width Glyph width (in pixels)

Usage

This section describes the process of extracting the data for a single glyph

1. Convert the glyph to an integer codepoint using `codepoint <- utf8ToInt(x)`
2. Use `row <- codepoint_to_idx[codepoint + 1]` to determine the row index of `glyph_info` which contains information for this codepoint.
3. if `row` is `NA` this indicates that the font does not support the glyph, and the `row` corresponding to `default_codepoint` should be used instead
4. `info <- glyph_info[row, ]`
5. Subset `coords` data.frame for the coordinates associated with this glyph: `coords[info$row_start:info$row_end, ]`

Examples

```
lifo <- get_lofi_font('unifont')
x <- 'a'
codepoint <- utf8ToInt(x)
row <- lifo$codepoint_to_idx[codepoint + 1]
info <- lifo$glyph_info[row,]
coords <- lifo$coords[seq(info$row_start, info$row_end), ]
coords
plot(coords$x, coords$y, asp = 1, ann = FALSE, axes = FALSE)

# Regular users should just use the functions provided in this package which
# add extra font information and layout sequences of characters
# over multiple lines
bitmap_text_coords('a', 'unifont')
bitmap_text_raster('a', 'unifont') |> plot()
```

---

plot.lofi-raster *Plot a lofi raster*

Description

Plot a lofi raster

Usage

```
## S3 method for class '`lofi-raster`'
plot(x, interpolate = FALSE, ...)
```

Arguments

x	lofi raster rendering
interpolate	default: FALSE
...	extra arguments passed to plot()

Value

None

Examples

```
ras <- bitmap_text_raster("Hi")
plot(ras)
```

---

print.lofi *Print summary information about a lofi font*

Description

Print summary information about a lofi font

Usage

```
## S3 method for class 'lofi'
print(x, ...)
```

Arguments

x	lofi font object
...	other arguments ignored

Value

None

Examples

```
font <- get_lofi_font('unscii-8')
print(font)
```

vector_text_coords *Create data.frame of glyph information for the given text.*

Description

Text input can contain multiple lines separated by carriage returns

Usage

```
vector_text_coords(
  text,
  font = c("gridfont_smooth", "gridfont", "arcade"),
  dx = 0L,
  dy = 0L,
  missing = utf8ToInt("?")
)
```

Arguments

text	Single text string. Can include carriage returns to split text over multiple lines.
font	Name of vector font, or a vector 'lofi' font object. Default: 'gridfont_smooth'. Use `get_lofi_names('vector')` to retrieve a list of all valid vector fonts included in this package.
dx	Additional character spacing in the horizontal direction. Default: 0
dy	Additional character spacing in the vertical direction i.e. between rows of text. Default: 0
missing	Codepoint to use if glyph not found in font. Default: NULL means to use the default specified by the font internally. Otherwise it will default to the codepoint for '?'

Value

data.frame of stroke information

`char_idx` The index of the character within the provided `text` string

`codepoint` Unicode codepoint (integer)

`stroke_idx` Index of the stroke within each character

`x` Pixel coordinate x value for display

`y` Pixel coordinate y value for display

`line` Line number within input `text` where this character appears

`x0` Original untransformed x-coordinate

`y0` Original untransformed y-coordinate

See Also

Other vector text functions: `vector_text_matrix()`, `vector_text_raster()`

Examples

```
vector_text_coords('Hi')
```

`vector_text_matrix` *Create a binary matrix of the rendered text*

Description

Create a binary matrix of the rendered text

Usage

```
vector_text_matrix(
  text,
  font = c("gridfont_smooth", "gridfont", "arcade"),
  scale_coords = 1,
  scale_matrix = 1,
  dx = NULL,
  dy = NULL,
  missing = utf8ToInt("?")
)
```

Arguments

`text`	Single text string. Can include carriage returns to split text over multiple lines.
`font`	Name of vector font, or a vector 'lofi' font object. Default: 'gridfont_smooth'. Use `get_lofi_names('vector')` to retrieve a list of all valid vector fonts included in this package.
`scale_coords`	Scale factor for text rendering. Numeric value greater than zero. Default: 1
`scale_matrix`	Integer size scale factor. Default: 1. Must be an integer value >= 1. Scale up the matrix or raster result by this factor after rendering the coordinates.
`dx`	Additional character spacing in the horizontal direction. Default: 0
`dy`	Additional character spacing in the vertical direction i.e. between rows of text. Default: 0
`missing`	Codepoint to use if glyph not found in font. Default: NULL means to use the default specified by the font internally. Otherwise it will default to the codepoint for '?'

Value

Binary matrix rendering of the font

See Also

Other vector text functions: `vector_text_coords()`, `vector_text_raster()`

Examples

```
vector_text_matrix("Hi")
```

`vector_text_raster` *Create a raster image of the rendered text*

Description

Create a raster image of the rendered text

Usage

```
vector_text_raster(
  text,
  font = c("gridfont_smooth", "gridfont", "arcade"),
  scale_coords = 10,
  scale_matrix = 1,
  dx = NULL,
  dy = NULL,
  missing = utf8ToInt("?")
)
```

Arguments

`text`	Single text string. Can include carriage returns to split text over multiple lines.
`font`	Name of vector font, or a vector 'lofi' font object. Default: 'gridfont_smooth'. Use `get_lofi_names('vector')` to retrieve a list of all valid vector fonts included in this package.
`scale_coords`	Scale factor for text rendering. Numeric value greater than zero. Default: 1
`scale_matrix`	Integer size scale factor. Default: 1. Must be an integer value >= 1. Scale up the matrix or raster result by this factor after rendering the coordinates.
`dx`	Additional character spacing in the horizontal direction. Default: 0
`dy`	Additional character spacing in the vertical direction i.e. between rows of text. Default: 0
`missing`	Codepoint to use if glyph not found in font. Default: NULL means to use the default specified by the font internally. Otherwise it will default to the codepoint for '?'

Value

Raster image of rendered text

See Also

Other vector text functions: vector_text_coords(), vector_text_matrix()

Examples

```
ras <- vector_text_raster("Hi")
plot(ras)
```

Index