Package 'fastpng'

February 27, 2026

Type Package

Title Read and Write PNG Files with Configurable Decoder/Encoder Options

Version 0.1.8

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description Read and write PNG images with arrays, rasters, native rasters, numeric arrays, integer arrays, raw vectors and indexed values. This PNG encoder exposes configurable internal options enabling the user to select a speed-size tradeoff. For example, disabling compression can speed up writing PNG by a factor of 50. Multiple image formats are supported including raster, native rasters, and integer and numeric arrays at color depths of 1, 2, 3 or 4. 16-bit images are also supported. This implementation uses the 'libspng' 'C' library which is available from <https://github.com/randy408/libspng/>.

License MIT + file LICENSE

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

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

LinkingTo colorfast

Imports colorfast (>= 1.0.1)

Depends R (>= 2.10)

Suggests knitr, png, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Copyright The included 'libspng' code is Copyright (c) 2018-2023, Randy <randy408@protonmail.com>. Sections within the 'libspng' code are derived from 'libpng' and copyright attributed to those authors. See 'COPYRIGHTS' for license information and full attribution of all copyrighted sections of code.

Encoding UTF-8

Language en-AU

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

NeedsCompilation yes

Author Mike Cheng [aut, cre, cph],
Randy408 [aut, cph] (Author of bundled libspng),
The PNG Reference Library Authors [aut, cph],
Cosmin Truta [cph] (SSE2 optimised filter functions, NEON optimised filter functions, NEON optimised palette expansion functions),
Glenn Randers-Pehrson [cph] (SSE2 optimised filter functions),
Andreas Dilger [cph],
Guy Eric Schalnat [cph],
Mike Klein [ctb] (SSE2 optimised filter functions),
Matt Sarett [ctb] (SSE2 optimised filter functions),
James Yu [ctb] (NEON optimised filter functions),
Mars Rullgard [ctb] (NEON optimised filter functions),
Arm Holdings [cph] (NEON optimised palette expansion functions),
Richard Townsend [ctb] (NEON optimised palette expansion functions)

Repository CRAN

Date/Publication 2026-02-27 22:50:02 UTC

Contents

`get_png_info` *Get information about a PNG file*

Description

Get information about a PNG file

Usage

```
get_png_info(src)
```

Arguments

src	PNG filename or raw vector containing PNG data

Value

Name list of information about the PNG image:

width,height Dimensions of PNG

bit_depth Bit depth. 8 or 16 bits

color_type,color_desc color type and its description

compression_method Compression setting

filter_method,filter_desc Filter method and description

interlace_method,interlace_desc Interlace method and description

Examples

```
# Create a small grayscale PNG image and fetch its PNG info
mat <- matrix(c(0L, 255L), 3, 4)
png_data <- write_png(mat)
get_png_info(png_data)
```

---

raw_spec *Create a specification for how raw bytes should be interpreted when passed to* write_png()

Description

Create a specification for how raw bytes should be interpreted when passed to write_png()

Usage

```
raw_spec(width, height, depth, bits)
```

Arguments

width, height	image dimensions
depth	number of colour channels. Integer value in range [1, 4]
bits	number of bits for each colour channel. Either 8 or 16.

Value

named list to pass to the write_png(..., raw_spec =)

Examples

```
raw_spec(100, 20, 3, 8)
```

read_png *Read a PNG*

Description

Read a PNG

Usage

```
read_png(
  src,
  type = c("array", "raster", "nativeraster", "indexed", "raw"),
  rgba = FALSE,
  flags = 1L,
  avoid_transpose = FALSE,
  array_type = c("dbl", "int")
)
```

Arguments

src	PNG image provided as either a file path, or a raw vector containing encoded PNG data
type	type of R object in which to store image data. Valid types are 'array', 'raster', "nativeraster", 'indexed' and 'raw'. Note that indexed image objects can only be loaded from indexed PNGs.
rgba	Should the result be forced into RGBA? Default: FALSE means to use the most appropriate format of the given R image type to store the data. If TRUE, then the image will be forced into RGBA color mode.
flags	Flags to apply when reading PNG. Default: 1 (always decode transparency from tRNS chunks). See ?spng_decode_flags for other options. Must be an integer.
avoid_transpose	Default: FALSE. If TRUE, then transposing the image from row-major (in the PNG), into column-major (in R) will be avoided if possible. This option only applies when reading grayscale or indexed images. Since the transposition is avoided, the decode step can be faster, but the image will not be in the correct orientation.
array_type	'dbl' or 'int'. Default: dbl. When reading PNG into an array, should the data be stored as a double (i.e. real) in the range [0, 1] or an integer in the range [0,255] (for 8 bit images) or [0,65535] (for 16 bit images).

Value

R image object of the specified type

Examples

```
# create a small greyscale matrix, and write it to a PNG file
ras <- matrix(c('#880000', '#000088'), 3, 4)
ras <- grDevices::as.raster(ras)
pngfile <- tempfile()
write_png(ras, file = pngfile)
ras2 <- read_png(pngfile, type = 'raster')
plot(ras2, interpolate = FALSE)
```

---

spng_decode_flags *Flags for decoding*

Description

SPNG_DECODE_TRNS Apply transparency

SPNG_DECODE_GAMMA Apply gamma correction

Usage

```
spng_decode_flags
```

Format

An object of class `list` of length 2.

---

test_image *Test images in various R formats*

Description

A nested named list of test images (300 x 200 pixels).

Usage

```
test_image
```

Format

An object of class `list` of length 9.

Details

Possible image color spaces within each image type

$gray Gray pixels representing intensity only

$gray_alpha Gray pixels with an alpha channel

$rgb RGB color image

$rgba RGB color image with alpha channel

A description of the image data within each image type

test_image$array Arrays of numeric values in the range [0, 1]

- **$gray** A 2D matrix
- **$gray_alpha** A 3D array with 2 planes i.e. `dim(x)[3] == 2`
- **$rgb** A 3D array with 3 planes i.e. `dim(x)[3] == 2`
- **$rgba** A 3D array with 4 planes i.e. `dim(x)[3] == 2`

test_image$array_16bit Same as `test_image$array` data except values contain 16 bits of signficant color information.

test_image$array_int Arrays of integer values in the range [0, 255]

- **$gray** A 2D matrix
- **$gray_alpha** A 3D array with 2 planes i.e. `dim(x)[3] == 2`
- **$rgb** A 3D array with 3 planes i.e. `dim(x)[3] == 2`
- **$rgba** A 3D array with 4 planes i.e. `dim(x)[3] == 2`

test_image$array_int_16bit Same as `test_image$array_int` data except values are in the range [0, 65535]

test_image$raster **$rgb** Raster image of color values given as hex codes `#RRGGBB`

- **$rgba** Raster image of color values given as hex codes `#RRGGBBAA`
- **$named** Raster image of color values given as R color names e.g. 'red', 'blue'

test_image$nativeraster Integer matrix of integer values. Each 32-bit numeric value holds a packed RGBA pixel

- **$rgba**

test_image$indexed **$integer_index** An integer matrix. Each value is an index into a separately specified color-lookup table

- **$numeric_index** A numeric matrix. Each value is an index into a separately specified color-lookup table
- **$palette** An example color palette to use with indexed images. 256 colors.

test_image$raw Sequences of raw bytes with attributes specifying 'width', 'height', 'depth' (i.e. number of colors) and 'bits' (number of bits for each color)

- **$gray** Sequence of gray pixels i.e. GGGG
- **$gray_alpha** Sequence of GA pixels i.e. GAGAGA...
- **$rgb** Sequence of RGB pixels i.e. RGBRGBRGB...
- **$rgba** Sequence of RGB pixels i.e. RGBARGBARGBA...

test_image$raw_16_bit The same as `test_image$raw` except each color takes 2 raw bytes.

`write_png` *Write PNG*

Description

Write PNG

Usage

```
write_png(
  image,
  file = NULL,
  palette = NULL,
  use_filter = TRUE,
  compression_level = -1L,
  avoid_transpose = FALSE,
  bits = 8,
  trns = NULL,
  raw_spec = NULL
)
```

Arguments

`image`	image. Supported image types: **Numeric arrays** Numeric arrays with values in the range [0, 1], with 1, 2, 3 or 4 colour planes to represent gray, gray+alpha, rgb and rgba pixels, respectively **Rasters** Rasters with a mixture of named colours (e.g. 'red'), and hex colours of the form #RGB, #RGBA, #RRGGBB and #RRGGBBAA **Integer arrays** Integer arrays with values in [0,255] treated as 8-bit image data. Integer arrays with values in [0, 65535] treated as 16-bit image data **Native rasters** Integer matrices containing colurs in native format i.e. 8-bit RGBA values packed into a single integer **Integer matrix + an indexed palette of colors** Can be saved as an indexed PNG **Raw vectors** Vectors of raw bytes must be accompanied by a `raw_spec` which details how the bytes are to be interpreted e.g. colour depth, width and height
`file`	If NULL (the default) then return PNG data as raw vector, otherwise write to the given file path.
`palette`	character vector of up to 256 colors. If specified, and the image is a 2D matrix of integer or numeric values, then an indexed PNG is written where the matrix values indicate the colour palette value to use. The values in the matrix must range from 0 (for the first colour)
`use_filter`	Use PNG filtering to help reduce size? Default: TRUE. If FALSE, then filtering will be disabled which can make image writing faster.

`compression_level`
: compression level for PNG. Default: -1 means to use the default compression level. Other valid values are in range [0, 9]. In general, lower compression levels result in faster compression, but larger image sizes. For fastest image writing, set `compression_level` to 0 to completely disable compression.

`avoid_transpose`
: Should transposition be avoided if possible so as to maximise the speed of writing the PNG? Default: FALSE. PNG is a row-major image format, but R stores data in column-major ordering. When writing data to PNG, it is often necessary to transpose the R data to match what PNG requires. If this option is set to `TRUE` then the image is written without this transposition and should speed up PNG creation. This option only has an effect for 2D integer and numeric matrix formats.

`bits`
: bit depth. default 8. Valid values are 8 and 16. This option only has an effect when image to output is a numeric array.

`trns`
: color to be treated as transparent in RGB and Greyscale images - without specifying a full alpha channel. Only a single color can be specified and it will be treated as a fully transparent color in the image. This setting is only used when writing RGB and Greyscale images. For 8-bit RGB images, the value may be a hex color value i.e. `"#RRGGBB"` or a vector of 3 numeric values in the range [0, 255]. For 8-bit greyscale images, must be a single integer value in the range [0, 255]. For 16-bit RGB images, the value may be a vector of 3 numeric values in the range [0, 65535]. For 16-bit greyscale images, must be a single integer value in the range [0, 65535]. Default: NULL - means to not add a transparency color.

`raw_spec`
: named list of image specifications for encoding a raw vector to PNG. Use `raw_spec()` to create such a list in the correct format. This argument is only required if the `image` argument is a raw vector.

Value

If `file` argument provided, function writes to file and returns nothing, otherwise it returns a raw vector holding the PNG encoded data.

Examples

```
# create a small greyscale integer matrix, and write it to a PNG file
mat <- matrix(c(0L, 255L), 3, 4)
pngfile <- tempfile()
write_png(mat, file = pngfile)
im <- read_png(pngfile, type = 'raster')
plot(im, interpolate = FALSE)
```

Index