Package 'png'

November 30, 2022

Version 0.1-8

Title Read and write PNG images

Author Simon Urbanek <Simon.Urbanek@r-project.org>

Maintainer Simon Urbanek <Simon.Urbanek@r-project.org>

Depends R (>= 2.9.0)

Description This package provides an easy and simple way to read, write and display bitmap images stored in the PNG format. It can read and write both files and in-memory raw vectors.

License GPL-2 | GPL-3

SystemRequirements libpng

URL http://www.rforge.net/png/

NeedsCompilation yes

R topics documented:

`readPNG` *Read a bitmap image stored in the PNG format*

Description

Reads an image from a PNG file/content into a raster array.

Usage

```
readPNG(source, native = FALSE, info = FALSE)
```

Arguments

source	Either name of the file to read from or a raw vector representing the PNG file content.
native	determines the image representation - if FALSE (the default) then the result is an array, if TRUE then the result is a native raster representation.
info	logical, if TRUE additional "info" attribute is attached to the result containing information from optional tags in the file (such as bit depth, resolution, gamma, text etc.). If the PNG file contains R metadata, it will also contain a "metadata" attribute with the unserialized R object.

Value

If native is FALSE then an array of the dimensions height x width x channels. If there is only one channel the result is a matrix. The values are reals between 0 and 1. If native is TRUE then an object of the class nativeRaster is returned instead. The latter cannot be easily computed on but is the most efficient way to draw using rasterImage.

Most common files decompress into RGB (3 channels), RGBA (4 channels), Grayscale (1 channel) or GA (2 channels). Note that G and GA images cannot be directly used in rasterImage unless native is set to TRUE because rasterImage requires RGB or RGBA format (nativeRaster is always 8-bit RGBA).

As of png 0.1-2 files with 16-bit channels are converted in full resolution to the array format, but the nativeRaster format only supports 8-bit and therefore a truncation is performed (eight least significant bits are dropped) with a warning if native is TRUE.

See Also

rasterImage, writePNG

Examples

```
# read a sample file (R logo)
img <- readPNG(system.file("img", "Rlogo.png", package="png"))

# read it also in native format
img.n <- readPNG(system.file("img", "Rlogo.png", package="png"), TRUE)

# if your R supports it, we'll plot it
if (exists("rasterImage")) { # can plot only in R 2.11.0 and higher
  plot(1:2, type='n')

  if (names(dev.cur()) == "windows") {
    # windows device doesn't support semi-transparency so we'll need
    # to flatten the image
    transparent <- img[,,4] == 0
    img <- as.raster(img[,,1:3])
    img[transparent] <- NA

    # interpolate must be FALSE on Windows, otherwise R will
    # try to interpolate transparency and fail
    rasterImage(img, 1.2, 1.27, 1.8, 1.73, interpolate=FALSE)

  } else {
    # any reasonable device will be fine using alpha
```

```
    rasterImage(img, 1.2, 1.27, 1.8, 1.73)
    rasterImage(img.n, 1.5, 1.5, 1.9, 1.8)

  }
}
```

---

writePNG *Write a bitmap image in PNG format*

Description

Create a PNG image from an array or matrix.

Usage

```
writePNG(image, target = raw(), dpi = NULL, asp = NULL,
         text = NULL, metadata = NULL)
```

Arguments

image	image represented by a real matrix or array with values in the range of 0 to 1. Values outside this range will be clipped. The object must be either two-dimensional (grayscale matrix) or three dimensional array (third dimension specifying the plane) and must have either one (grayscale), two (grayscale + alpha), three (RGB) or four (RGB + alpha) planes. (For alternative image specifications see details)
target	Either name of the file to write, a binary connection or a raw vector (`raw()` - the default - is good enough) indicating that the output should be a raw vector.
dpi	optional, if set, must be a numeric vector of length 1 or 2 specifying the resolution of the image in DPI (dots per inch) for x and y (in that order) - it is recycled to length 2.
asp	optional, if set, must be a numeric scalar specifying the aspect ratio (`x / y`). `dpi` and `asp` are mutually exclusive, speciyfing both is an error.
text	optional, named character vector of entries that will be saved in the text chunk of the PNG. Names are used as keys. Note that the `"R.metadata"` key is reserved for internal use - see below
metadata	optional, an R object that will be serialized into the `"R.metadata"` text key

Details

`writePNG` takes an image as input and compresses it into PNG format. The image input is usually a matrix (for grayscale images - dimensions are width, height) or an array (for color and alpha images - dimensions are width, height, planes) of reals. The planes are interpreted in the sequence red, green, blue, alpha.

Alternative representation of an image is of `nativeRaster` class which is an integer matrix with each entry representing one pixel in binary encoded RGBA format (as used internally by R). It can be obtained from `readPNG` using `native = TRUE`.

Finally, `writePNG` also supports raw array containing the RGBA image as bytes. The dimensions of the raw array have to be planes, width, height (because the storage is interleaved). Currently only 4 planes (RGBA) are supported and the processing is equivalent to that of a native raster.

The result is either stored in a file (if `target` is a file name), in a raw vector (if `target` is a raw vector) or sent to a binary connection.

If either `dpi` or `asp` is set, the `sPHy` chunk is generated based on that information. Note that not all image viewers interpret this setting, and even fewer support non-square pixels.

Value

Either `NULL` if the target is a file or a raw vector containing the compressed PNG image if the target was a raw vector.

Note

Currently `writePNG` only produces 8-bit, deflate-compressed, non-quantized, non-interlaced images. Note in particular that `readPNG` can read 16-bit channels but storing them back using `writePNG` will strip the 8 LSB (irrelevant for display purposes but possibly relevant for use of PNG in signal-processing if the input is truly 16-bit wide).

Author(s)

Simon Urbanek

See Also

`readPNG`

Examples

```
# read a sample file (R logo)
img <- readPNG(system.file("img","Rlogo.png",package="png"))
# write the image into a raw vector
r <- writePNG(img)
# read it back again
img2 <- readPNG(r)
# it better be the same
identical(img, img2)
# try to write a native raster
img3 <- readPNG(system.file("img","Rlogo.png",package="png"), TRUE)
r2 <- writePNG(img3)
img4 <- readPNG(r2, TRUE)
identical(img3, img4)

## text and metadata
r <- writePNG(img, text=c(source=R.version.string),
     metadata=sessionInfo())
img5 <- readPNG(r, info=TRUE)
attr(img5, "info")
attr(img5, "metadata")
```

Index