Package: tiff (via r-universe)

October 28, 2026

Version 0.1-12

Title Read and Write TIFF Images

Author Simon Urbanek <Simon.Urbanek@r-project.org> [aut, cre], Kent Johnson <kjohnson@akoyabio.com> [ctb]

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

Depends R (>= 2.9.0)

Description Functions to read, write and display bitmap images stored in the TIFF format. It can read and write both files and in-memory raw vectors, including native image representation.

License GPL-2 | GPL-3

SystemRequirements tiff and jpeg libraries

URL https://www.rforge.net/tiff/

NeedsCompilation yes

Config/pak/sysreqs libjpeg-dev libtiff-dev

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

Date/Publication 2023-11-29 02:48:36 UTC

RemoteUrl https://github.com/cran/tiff

RemoteRef HEAD

RemoteSha 84d98817986360dc5a9aacf285be301522fb37bf

Contents

readTIFF 2
writeTIFF 4

Index **6**

readTIFF *Read a bitmap image stored in the TIFF format*

Description

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

Usage

```
readTIFF(source, native = FALSE, all = FALSE, convert = FALSE,
         info = FALSE, indexed = FALSE, as.is = FALSE,
  payload = TRUE)
```

Arguments

source	Either name of the file to read from or a raw vector representing the TIFF file content.
native	logical, determines the image representation - if FALSE (the default) then the result is an array, if TRUE then the result is a native raster representation (suitable for plotting).
all	logical scalar or integer vector. TIFF files can contain more than one image. If all=TRUE then all images are returned in a list of images. If all is a vector, it gives the (1-based) indices of images to return. Otherwise only the first image is returned.
convert	logical, if TRUE then first convert the image into 8-bit RGBA samples and then to an array, see below for details.
info	logical, if set to TRUE then the resulting image(s) will also contain information from TIFF tags as attributes
indexed	logical, if set to TRUE then indexed images will be returned in the indexed form, i.e., as a matrix of integer indices referencing into a color map which is returned in the "color.map" attribute. This flag cannot be combined with convert or native and has no effect on images that are not indexed.
as.is	logical, if TRUE an attempt will be made to return the original integer values without re-scaling where possible
payload	logical, if FALSE then only metadata about the image(s) is returned, but not the actual image. Implies info=TRUE and all image-related flags are ignored.

Details

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).

TIFF images can have a wide range of internal representations, but only the most common in image processing are directly supported (8-bit, 16-bit integer and 32-bit float samples). Other formats

(color maps, sub-8-bit images, etc.) are only supported via `convert=TRUE` which uses the built-in facilities of the TIFF library to convert the image into RGBA format with 8-bit samples (i.e. total of 32-bit per pixel) and then store the relevant components from there into real arrays. This is the same path as used by `native=TRUE` and so differs only in the output value. Note that conversion may result in different values than direct acccess as it is intended mainly for viewing and not computation.

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 (except for 32-bit floating point sample storage which are unscaled reals, and for indexed and `as.is=TRUE` which are integers). 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`.

If `all` is `TRUE` or a vector of image indices, then the result is a list of the above with zero or more elements. If `all` is a vector of indices, the result will have exactly the same length as `all`. If an index does not appear in the file, the corresponding list entry will be `NULL`.

If `payload=FALSE` then the result is equivalent to `info=TRUE` but without the image data and returned as a data frame. If `all` is either `TRUE` or a vector then the result will be a data frame with each row corresponding to one image.

Note

Some non-standard formats such as 12-bit TIFFs are partially supported (there is no standard for packing order for TIFFs beoynd 8-bit so we assume big-endian packing similar to the default fill order and only support single channel or indexed).

The `as.is=TRUE` option is experimental, cannot be used with `native` or `convert` and only works for integer storage TIFFs.

Author(s)

Simon Urbanek Kent Johnson

See Also

`rasterImage`, `writeTIFF`

Examples

```
Rlogo <- system.file("img", "Rlogo.tiff", package="tiff")

# read a sample file (R logo)
img <- readTIFF(Rlogo)

# read it also in native format
img.n <- readTIFF(Rlogo, native=TRUE)

# and also in converted
img.c <- readTIFF(Rlogo, convert=TRUE)
```

```
# read all contained images
str(readTIFF(Rlogo, all=TRUE))

# pick some images
str(readTIFF(Rlogo, all=c(5, 1, 3)))

# only show information
str(readTIFF(Rlogo, payload=FALSE))

# 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)
  }
}
```

`writeTIFF` *Write one or more bitmap images in TIFF format*

Description

Writes images into a TIFF file or a raw vector representing such.

Usage

```
writeTIFF(what, where, bits.per.sample = 8L,
          compression = c("LZW", "none", "PackBits", "RLE", "JPEG", "deflate"),
          reduce = TRUE)
```

Arguments

`what`	either an image or a list of images. An image is a real matrix or array of three dimensions, or an object of the class `"nativeRaster"`.
`where`	file name or a raw vector

`bits.per.sample`
: number of bits per sample (numeric scalar). Supported values in this version are 8, 16, and 32.

`compression`
: desired compression algorithm (string). Optionally, it can be specified as a numeric value corresponding to the compression TIFF tag, but it needs to be also supported by the underlying TIFF library

`reduce`
: if `TRUE` then `writeTIFF` will attempt to reduce the number of planes in native rasters by analyzing the image to choose one of RGBA, RGB, GA or G formats, whichever uses the least planes without any loss. Otherwise the image is always saved with four planes (RGBA).

Details

By default `writeTIFF` uses the same number of planes as there are planes in the input image. For native images it is always four unless `reduce = TRUE` is set (see above). Consequently, color maps are not used. The output always uses contiguous planar configuration (baseline TIFF). The output is tagged with a photometric tag of either RGB (3 or 4 planes) or zero-is-black (1 or 2 planes). If `what` is a list then the TIFF output will be a directory of the corresponding number of images (in TIFF speak - not to be confused with file directories).

Value

If `where` is a raw vector then the value is the raw vector containg the TIFF contents, otherwise a scalar integer specifying the number of images written in the file.

Author(s)

Simon Urbanek

See Also

`readTIFF`

Examples

```
img <- readTIFF(system.file("img", "Rlogo.tiff", package="tiff"))
# write without the alpha channel
tiff <- writeTIFF(img[,,-4], raw(0))
# read as native
i2 <- readTIFF(tiff, native=TRUE)
# write reduced - should be the same as tiff
t2 <- writeTIFF(i2, raw(0), reduce=TRUE)
```

Index

∗ **IO**
    readTIFF, 2
    writeTIFF, 4

rasterImage, *2, 3*
readTIFF, 2, *5*

writeTIFF, *3*, 4