Package 'ijtiff'

January 29, 2025

Type Package

Title Comprehensive TIFF I/O with Full Support for 'ImageJ' TIFF Files

Version 2.3.5

Maintainer Rory Nolan <rorynoolan@gmail.com>

Description General purpose TIFF file I/O for R users. Currently the
only such package with read and write support for TIFF files with
floating point (real-numbered) pixels, and the only package that can
correctly import TIFF files that were saved from 'ImageJ' and write
TIFF files than can be correctly read by 'ImageJ'
<https://imagej.net/ij/>. Also supports text image I/O.

License GPL-3

URL https://docs.ropensci.org/ijtiff/,
https://github.com/ropensci/ijtiff

BugReports https://github.com/ropensci/ijtiff/issues

Depends R (>= 3.5)

Imports checkmate (>= 1.9.3), cli, dplyr, fs (>= 1.5), graphics,
imager, magrittr (>= 1.5), methods, purrr, readr, rlang (>=
1.0), strex (>= 1.5), stringr (>= 1.4), withr (>= 2.1), zeallot

Suggests abind, covr, EBImage, knitr, magick, mockery, pacman,
rmarkdown, rprojroot, spelling, testthat (>= 3.0), tiff, vdiffr

VignetteBuilder knitr

Biarch TRUE

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

SystemRequirements fftw3, libtiff, libbz2, libdeflate, libjpeg,
liblzma, libwebp, libzstd, X11, zlib

NeedsCompilation yes

Author Rory Nolan [aut, cre] (<<https://orcid.org/0000-0002-5239-4043>>),
      Kent Johnson [aut],
      Simon Urbanek [ctb],
      Sergi Padilla-Parra [ths] (<<https://orcid.org/0000-0002-8010-9481>>),
      Jeroen Ooms [rev, ctb] (<<https://orcid.org/0000-0002-4035-0289>>),
      Jon Clayden [rev] (<<https://orcid.org/0000-0002-6608-0619>>)

Contents

---

as_EBImage   *Convert an ijtiff_img to an EBImage::Image.*

Description

This is for interoperability with the the EBImage package.

Usage

```
as_EBImage(img, colormode = NULL, scale = TRUE, force = TRUE)
```

Arguments

img   An ijtiff_img object (or something coercible to one).

colormode   A numeric or a character string containing the color mode which can be either
              "Grayscale" or "Color". If not specified, a guess is made. See 'Details'.

scale   Scale values in an integer image to the range [0, 1]? Has no effect on floating-
              point images.

force   This function is designed to take ijtiff_imgs as input. To force any old array
              through this function, use force = TRUE, but take care to check that the result is
              what you'd like it to be.

Details

The guess for the `colormode` is made as follows: * If `img` has an attribute `color_space` with value
`"RGB"`, then `colormode` is set to `"Color"`. * Else if `img` has 3 or 4 channels, then `colormode` is set
to `"Color"`. * Else `colormode` is set to "Grayscale".

Value

An [EBImage::Image](#).

Examples

```
if (rlang::is_installed("EBImage")) {
  img <- read_tif(system.file("img", "Rlogo.tif", package = "ijtiff"))
  str(img)
  str(as_EBImage(img))
}
```

---

count_frames   *Count the number of frames in a TIFF file.*

Description

TIFF files can hold many frames. Often this is sensible, e.g. each frame could be a time-point in a
video or a slice of a z-stack.

Usage

```
count_frames(path)

frames_count(path)
```

Arguments

path   A string. The path to the tiff file to read.

Details

For those familiar with TIFF files, this function counts the number of directories in a TIFF file.
There is an adjustment made for some ImageJ-written TIFF files.

Value

A number, the number of frames in the TIFF file. This has an attribute `n_dirs` which holds the true
number of directories in the TIFF file, making no allowance for the way ImageJ may write TIFF
files.

Examples

```
count_frames(system.file("img", "Rlogo.tif", package = "ijtiff"))
```

---

display *Basic image display.*

Description

Display an image that has been read in by read_tif() as it would look in 'ImageJ'. This function is really just imager's plot.cimg() on the inside, with some handling.

Usage

```
display(img, ...)
```

Arguments

img	An ijtiff_img object.
...	Passed to imager's 'plot.cimg().

Examples

```
img <- read_tif(system.file("img", "Rlogo.tif", package = "ijtiff"))
display(img)
display(img[, , 1, 1]) # first (red) channel, first frame
display(img[, , 2, ]) # second (green) channel, first frame
display(img[, , 3, ]) # third (blue) channel, first frame
display(img, basic = TRUE) # displays first (red) channel, first frame
```

---

ijtiff ijtiff*: TIFF I/O for* ImageJ *users*

Description

This is a general purpose TIFF I/O utility for R. The tiff package already exists for this purpose but ijtiff adds some functionality and overcomes some bugs therein.

Details

- ijtiff can write TIFF files whose pixel values are real (floating-point) numbers; tiff cannot.
- ijtiff can read and write *text images*; tiff cannot.
- tiff struggles to interpret channel information and gives cryptic errors when reading TIFF files written by the *ImageJ* software; ijtiff works smoothly with these images.

Author(s)

Maintainer: Rory Nolan <rorynoolan@gmail.com> ([ORCID](#))

Authors:

- Kent Johnson <kjohnson@akoyabio.com>

Other contributors:

- Simon Urbanek <Simon.Urbanek@r-project.org> [contributor]
- Sergi Padilla-Parra <spadilla@well.ox.ac.uk> ([ORCID](#)) [thesis advisor]
- Jeroen Ooms ([ORCID](#)) [reviewer, contributor]
- Jon Clayden ([ORCID](#)) [reviewer]

See Also

Useful links:

- <https://docs.ropensci.org/ijtiff/>
- <https://github.com/ropensci/ijtiff>
- Report bugs at <https://github.com/ropensci/ijtiff/issues>

---

ijtiff_img	ijtiff_img *class.*

Description

A class for images which are read or to be written by the `ijtiff` package.

Usage

```
ijtiff_img(img, ...)

as_ijtiff_img(img, ...)
```

Arguments

img   An array representing the image.

- For a single-plane, grayscale image, use a matrix `img[y, x]`.
- For a multi-plane, grayscale image, use a 3-dimensional array `img[y, x, plane]`.
- For a multi-channel, single-plane image, use a 4-dimensional array with a redundant 4th slot `img[y, x, channel, ]` (see [ijtiff_img](#) 'Examples' for an example).
- For a multi-channel, multi-plane image, use a 4-dimensional array `img[y, x, channel, plane]`.

...   Named arguments which are set as attributes.

Value

A 4 dimensional array representing an image, indexed by img[y, x, channel, frame], with se-
lected attributes.

Examples

```
img <- matrix(1:4, nrow = 2) # to be a single-channel, grayscale image
ijtiff_img(img, description = "single-channel, grayscale")
img <- array(seq_len(2^3), dim = rep(2, 3)) # 1 channel, 2 frame
ijtiff_img(img, description = "blah blah blah")
img <- array(seq_len(2^3), dim = c(2, 2, 2, 1)) #  2 channel, 1 frame
ijtiff_img(img, description = "blah blah")
img <- array(seq_len(2^4), dim = rep(2, 4)) # 2 channel, 2 frame
ijtiff_img(img, software = "R")
```

---

linescan-conversion   *Rejig linescan images.*

Description

ijtiff has the fourth dimension of an [ijtiff_img](#) as its time dimension. However, some linescan
images (images where a single line of pixels is acquired over and over) have the time dimension
as the y dimension, (to avoid the need for an image stack). These functions allow one to convert
this type of image into a conventional [ijtiff_img](#) (with time in the fourth dimension) and to convert
back.

Usage

```
linescan_to_stack(linescan_img)

stack_to_linescan(img)
```

Arguments

linescan_img	A 4-dimensional array in which the time axis is the first axis. Dimension 4 must be 1 i.e. dim(linescan_img)[4] == 1.
img	A conventional [ijtiff_img](#), to be turned into a linescan image. Dimension 1 must be 1 i.e. dim(img)[1] == 1.

Value

The converted image, an object of class [ijtiff_img](#).

Examples

```
linescan <- ijtiff_img(array(rep(1:4, each = 4), dim = c(4, 4, 1, 1)))
print(linescan)
stack <- linescan_to_stack(linescan)
print(stack)
linescan <- stack_to_linescan(stack)
print(linescan)
```

---

print.ijtiff_img	*Print method for an* ijtiff_img.

Description

Print method for an ijtiff_img.

Usage

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

Arguments

x	An object of class ijtiff_img.
...	Not currently used.

Value

The input (invisibly).

---

read_tags	*Read TIFF tag information without actually reading the image array.*

Description

TIFF files contain metadata about images in their *TIFF tags*. This function is for reading this information without reading the actual image.

Usage

```
read_tags(path, frames = 1)

tags_read(path, frames = 1)
```

Arguments

path	A string. The path to the tiff file to read.
frames	Which frames do you want to read tags from. Default first frame only. To read from the 2nd and 7th frames, use `frames = c(2, 7)`, to read from all frames, use `frames = "all"`.

Value

A list of lists.

Author(s)

Simon Urbanek, Kent Johnson, Rory Nolan.

See Also

[read_tif()](read_tif)

Examples

```
read_tags(system.file("img", "Rlogo.tif", package = "ijtiff"))
read_tags(system.file("img", "Rlogo-banana.tif", package = "ijtiff"),
  frames = c(2, 4)
)
```

---

read_tif	*Read an image stored in the TIFF format*

Description

Reads an image from a TIFF file/content into a numeric array or list.

Usage

```
read_tif(path, frames = "all", list_safety = "error", msg = TRUE)

tif_read(path, frames = "all", list_safety = "error", msg = TRUE)
```

Arguments

path	A string. The path to the tiff file to read.
frames	Which frames do you want to read. Default all. To read the 2nd and 7th frames, use `frames = c(2, 7)`.
list_safety	A string. This is for type safety of this function. Since returning a list is unlikely and probably unexpected, the default is to error. You can instead opt to throw a warning (`list_safety = "warning"`) or to just return the list quietly (`list_safety = "none"`).
msg	Print an informative message about the image being read?

Details

TIFF files have the capability to store multiple images, each having multiple channels. Typically, these multiple images represent the sequential frames in a time-stack or z-stack of images and hence each of these images has the same dimension. If this is the case, they are all read into a single 4-dimensional array `img` where `img` is indexed as `img[y, x, channel, frame]` (where we have `y, x` to comply with the conventional `row, col` indexing of a matrix - it means that images displayed as arrays of numbers in the R console will have the correct orientation). However, it is possible that the images in the TIFF file have varying dimensions (most people have never seen this), in which case they are read in as a list of images, where again each element of the list is a 4-dimensional array `img`, indexed as `img[y, x, channel, frame]`.

A (somewhat random) set of TIFF tags are attributed to the read image. These are IMAGEDEPTH, BITSPERSAMPLE, SAMPLESPERPIXEL, SAMPLEFORMAT, PLANARCONFIG, COMPRESSION, THRESHHOLDING, XRESOLUTION, YRESOLUTION, RESOLUTIONUNIT, INDEXED and ORIENTATION. More tags should be added in a subsequent version of this package. You can read about TIFF tags at https://www.awaresystems.be/imaging/tiff/tifftags.html.

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

Value

An object of class ijtiff_img or a list of ijtiff_imgs.

Note

- 12-bit TIFFs are not supported.
- There is no standard for packing order for TIFFs beyond 8-bit so we assume big-endian packing

.

Author(s)

Simon Urbanek wrote most of this code for the 'tiff' package. Rory Nolan lifted it from there and changed it around a bit for this 'ijtiff' package. Credit should be directed towards Lord Urbanek.

See Also

write_tif()

Examples

```
img <- read_tif(system.file("img", "Rlogo.tif", package = "ijtiff"))
```

---

text-image-io   *Read/write an image array to/from disk as text file(s).*

Description

Write images (arrays) as tab-separated `.txt` files on disk. Each channel-frame pair gets its own file.

Usage

```
write_txt_img(img, path, rds = FALSE, msg = TRUE)

read_txt_img(path, msg = TRUE)

txt_img_write(img, path, rds = FALSE, msg = TRUE)

txt_img_read(path, msg = TRUE)
```

Arguments

img	An image, represented by a 4-dimensional array, like an ijtiff_img.
path	The name of the input/output output file(s), *without* a file extension.
rds	In addition to writing a text file, save the image as an RDS (a single R object) file?
msg	Print an informative message about the image being read?

Examples

```
img <- read_tif(system.file("img", "Rlogo.tif", package = "ijtiff"))
tmptxt <- tempfile(pattern = "img", fileext = ".txt")
write_txt_img(img, tmptxt)
tmptxt_ch1_path <- paste0(strex::str_before_last_dot(tmptxt), "_ch1.txt")
print(tmptxt_ch1_path)
txt_img <- read_txt_img(tmptxt_ch1_path)
```

---

tif_tags_reference   *TIFF tag reference.*

Description

A dataset containing the information on all known baseline and extended TIFF tags.

Usage

```
tif_tags_reference()
```

Details

A data frame with 96 rows and 10 variables:

code_dec decimal numeric code of the TIFF tag

code_hex hexadecimal numeric code of the TIFF tag

name the name of the TIFF tag

short_description a short description of the TIFF tag

tag_type the type of TIFF tag: either "baseline" or "extended"

url the URL of the TIFF tag at <https://www.awaresystems.be>

libtiff_name the TIFF tag name in the libtiff C library

c_type the C type of the TIFF tag data in libtiff

count the number of elements in the TIFF tag data

default the default value of the data held in the TIFF tag

Source

<https://www.awaresystems.be>

Examples

```
tif_tags_reference()
```

---

write_tif	*Write images in TIFF format*

Description

Write images into a TIFF file.

Usage

```
write_tif(
  img,
  path,
  bits_per_sample = "auto",
  compression = "none",
  overwrite = FALSE,
  msg = TRUE
)

tif_write(
  img,
  path,
  bits_per_sample = "auto",
```

```
    compression = "none",
    overwrite = FALSE,
    msg = TRUE
)
```

Arguments

img   An array representing the image.

- For a single-plane, grayscale image, use a matrix img[y, x].
- For a multi-plane, grayscale image, use a 3-dimensional array img[y, x, plane].
- For a multi-channel, single-plane image, use a 4-dimensional array with a redundant 4th slot img[y, x, channel,] (see ijtiff_img 'Examples' for an example).
- For a multi-channel, multi-plane image, use a 4-dimensional array img[y, x, channel, plane].

path   file name or a raw vector

bits_per_sample

number of bits per sample (numeric scalar). Supported values are 8, 16, and 32. The default "auto" automatically picks the smallest workable value based on the maximum element in img. For example, if the maximum element in img is 789, then 16-bit will be chosen because 789 is greater than $2 \char`\^ 8$ - 1 but less than or equal to $2 \char`\^ 16$ - 1.

compression   A string, the desired compression algorithm. Must be one of "none", "LZW", "PackBits", "RLE", "JPEG", "deflate" or "Zip". If you want compression but don't know which one to go for, I recommend "Zip", it gives a large file size reduction and it's lossless. Note that "deflate" and "Zip" are the same thing. Avoid using "JPEG" compression in a TIFF file if you can; I've noticed it can be buggy.

overwrite   If writing the image would overwrite a file, do you want to proceed?

msg   Print an informative message about the image being written?

Value

The input img (invisibly).

Author(s)

Simon Urbanek wrote most of this code for the 'tiff' package. Rory Nolan lifted it from there and changed it around a bit for this 'ijtiff' package. Credit should be directed towards Lord Urbanek.

See Also

read_tif()

Examples

```
img <- read_tif(system.file("img", "Rlogo.tif", package = "ijtiff"))
temp_dir <- tempdir()
write_tif(img, paste0(temp_dir, "/", "Rlogo"))
img <- matrix(1:4, nrow = 2)
write_tif(img, paste0(temp_dir, "/", "tiny2x2"))
list.files(temp_dir, pattern = "tif$")
```

Index