Package 'rasterpic'

March 19, 2025

Title Convert Digital Images into 'SpatRaster' Objects

Version 0.3.0

Description Generate 'SpatRaster' objects, as defined by the 'terra' package, from digital images, using a specified spatial object as a geographical reference.

License MIT + file LICENSE

URL https://dieghernan.github.io/rasterpic/, https://github.com/dieghernan/rasterpic

BugReports https://github.com/dieghernan/rasterpic/issues

Depends R (>= 3.6.0)

Imports png (>= 0.1-5), sf (>= 1.0.0), terra (>= 1.4-22)

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

VignetteBuilder knitr

Config/Needs/check curl

Config/Needs/coverage curl

Config/Needs/website dieghernan/gitdevr, tmap, mapsf, maptiles, devtools, curl, remotes, cpp11

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.2

X-schema.org-keywords cran, jpeg, jpg, maps, png, r, r-package, r-stats, raster, rstats, sf, terra, tif, tiff, cran-r

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph] (<https://orcid.org/0000-0001-8457-4658>)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2025-03-19 17:50:02 UTC

Contents

`rasterpic_img` *Convert an image to a geo-tagged* `SpatRaster`

Description

Geotags an image based on the coordinates of a given spatial object.

Usage

```
rasterpic_img(
  x,
  img,
  halign = 0.5,
  valign = 0.5,
  expand = 0,
  crop = FALSE,
  mask = FALSE,
  inverse = FALSE,
  crs = NULL
)
```

Arguments

`x`	**R** object that may be: • An object created with **sf** of class `sf`, `sfc`, `sfg` or `bbox`). • An object created with **terra** of class `SpatRaster`, `SpatVector` or `SpatExtent`. • A numeric vector of length 4 with the extent to be used for geotagging (i.e. `c(xmin, ymin, xmax, ymax)`).
`img`	An image to be geotagged. It can be a local file or an online file (e.g. `"https://i.imgur.com/6yHmlwT.j` The following image extensions are accepted: • `png`. • `jpeg/jpg`. • `tiff/tif`.
`halign, valign`	Horizontal and vertical alignment of `img` with respect to `x`. It should be a value between `0` and `1`: • `halign = 0, valign = 0` assumes that `x` should be in the bottom left corner of the `SpatRaster`. • `halign = 1, valign = 1` assumes that `x` should be in the top right corner of the `SpatRaster`.

- The default `halign = .5, valign = .5` assumes that `x` is the center of `img`. See `vignette("rasterpic", package = "rasterpic")` for examples.

`expand` An expansion factor of the bounding box of `x`. `0` means that no expansion is added, `1` means that the bounding box is expanded to double the original size. See **Details**.

`crop` Logical. Should the raster be cropped to the (expanded) bounding box of `x`? See **Details**.

`mask` Logical, applicable only if `x` is a `sf`, `sfc` or `SpatVector` object. Should the raster be masked to `x`? See **Details**.

`inverse` Logical. It affects only if `mask = TRUE`. If `TRUE`, areas on the raster that do not overlap with `x` are masked.

`crs` Character string describing a coordinate reference system. This parameter would only affect if `x` is a `SpatExtent`, `sfg`, `bbox` or a vector of coordinates. See **CRS** section.

Details

`vignette("rasterpic", package = "rasterpic")` explains with examples the effect of parameters `halign`, `valign`, `expand`, `crop` and `mask`.

CRS:

The function preserves the Coordinate Reference System of `x` if applicable. For optimal results **do not use** geographic coordinates (longitude/latitude).

`crs` can be in a WKT format, as a `"authority:number"` code such as `"EPSG:4326"`, or a PROJ-string format such as `"+proj=utm +zone=12"`. It can be also retrieved with:

- `sf::st_crs(25830)$wkt`.
- `terra::crs()`.
- `tidyterra::pull_crs()`.

See **Value** and **Notes** on `terra::crs()`.

Value

A `SpatRaster` object (see `terra::rast()`) where each layer corresponds to a color channel of `img`:

- If `img` has at least 3 channels (e.g. layers), the result would have an additional property setting the layers 1 to 3 as the Red, Green and Blue channels.
- If `img` already has a definition or RGB values (this may be the case for `tiff/tif` files) the result would keep that channel definition.

See Also

From **sf**:

- `sf::st_crs()`.
- `sf::st_bbox()`.
- `vignette("sf1", package = "sf")` to understand how **sf** organizes **R** objects.

From **terra**:

- `terra::vect()`, `terra::rast()` and `terra::ext()`.
- `terra::mask()`.
- `terra::crs()`.
- `terra::RGB()`.

For plotting:

- `terra::plot()` and `terra::plotRGB()`.
- With **ggplot2** use **tidyterra**:
  - `tidyterra::autoplot.SpatRaster()`.
  - `tidyterra::geom_spatraster_rgb()`.

Examples

```
library(sf)
library(terra)
library(ggplot2)
library(tidyterra)


x_path <- system.file("gpkg/UK.gpkg", package = "rasterpic")
x <- st_read(x_path, quiet = TRUE)
img <- system.file("img/vertical.png", package = "rasterpic")

# Default config
ex1 <- rasterpic_img(x, img)

ex1

autoplot(ex1) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5)


# Expand
ex2 <- rasterpic_img(x, img, expand = 0.5)

autoplot(ex2) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5)


# Align
ex3 <- rasterpic_img(x, img, halign = 0)

autoplot(ex3) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5)
labs(title = "Align")

# Crop
ex4 <- rasterpic_img(x, img, crop = TRUE)
```

```
autoplot(ex4) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5) +
  labs(title = "Crop")

# Mask
ex5 <- rasterpic_img(x, img, mask = TRUE)

autoplot(ex5) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5) +
  labs(title = "Mask")

# Mask inverse
ex6 <- rasterpic_img(x, img, mask = TRUE, inverse = TRUE)

autoplot(ex6) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5) +
  labs(title = "Mask Inverse")

# Combine Mask inverse and crop
ex7 <- rasterpic_img(x, img, crop = TRUE, mask = TRUE, inverse = TRUE)

autoplot(ex7) +
  geom_sf(data = x, fill = NA, color = "white", linewidth = .5) +
  labs(title = "Combine")

# RGB channels ------
plot(ex1)
ex_rgb <- ex1
has.RGB(ex_rgb)
RGB(ex_rgb)

# Modify RGB channels
RGB(ex_rgb) <- c(2, 3, 1)
RGB(ex_rgb)

plot(ex_rgb)

# Remove RGB channels
RGB(ex_rgb) <- NULL
has.RGB(ex_rgb)
RGB(ex_rgb)

# Note the difference with terra::plot
plot(ex_rgb)
```

Index