

Package ‘rasterpic’

January 15, 2026

Title Convert Digital Images into 'SpatRaster' Objects

Version 0.3.1

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 (>= 4.1.0)

Imports png, sf (>= 1.0.0), terra (>= 1.8-21)

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 7.3.3

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] (ORCID:
<<https://orcid.org/0000-0001-8457-4658>>)

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

Repository CRAN

Date/Publication 2026-01-15 00:10:02 UTC

Contents

rasterpic_img	2
Index	7

rasterpic_img	<i>Convert an image to a geo-tagged SpatRaster</i>
---------------	--

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. <code>c(xmin, ymin, xmax, ymax)</code>).
img	An image to be geotagged. It can be a local file or an online file (e.g. "<code>https://i.imgur.com/6yHmlwT.j</code>"). The following image extensions are accepted: • png. • jpeg/jpg. • tiff/tif.
halign, valign	Horizontal and vertical alignment of <code>img</code> with respect to <code>x</code>. They should be values between 0 and 1: • <code>halign = 0</code>, <code>valign = 0</code> assumes that <code>x</code> should be in the bottom left corner of the SpatRaster. • <code>halign = 1</code>, <code>valign = 1</code> assumes that <code>x</code> should be in the top right corner of the SpatRaster.

	The default <code>halign = .5</code>, <code>valign = .5</code> assumes that <code>x</code> is the center of <code>img</code>. See <code>vignette("rasterpic", package = "rasterpic")</code> for examples.
<code>expand</code>	An expansion factor of the bounding box of <code>x</code> . <code>0</code> means that no expansion is added, <code>1</code> means that the bounding box is expanded to double the original size. See Details .
<code>crop</code>	Logical. Should the raster be cropped to the (expanded) bounding box of <code>x</code> ? See Details .
<code>mask</code>	Logical, applicable only if <code>x</code> is a <code>sf</code> , <code>sfc</code> or <code>SpatVector</code> object. Should the raster be masked to <code>x</code> ? See Details .
<code>inverse</code>	Logical. It only affects when <code>mask = TRUE</code> . If <code>TRUE</code> , areas on the raster that do not overlap with <code>x</code> are masked.
<code>crs</code>	Character string describing a coordinate reference system. This parameter only affects when <code>x</code> is a <code>SpatExtent</code> , <code>sfg</code> , <code>bbox</code> 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 also be 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 will have an additional property setting the layers 1 to 3 as the Red, Green and Blue channels with names "r" "g" "b" and alpha (if applicable).
- If `img` already has a definition or RGB values (this may be the case for `tiff/tif` files) the result will keep that channel definition.

The resulting `SpatRaster` will have an RGB specification as explained in `terra::RGB()`.

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()`.
- Other packages:
 - **tmap**.
 - **mapsf**.
 - **maptiles**.

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

bbox, [2](#)

masked, [3](#)

rasterpic_img, [2](#)

sf, [2](#)

sf::st_bbox(), [4](#)

sf::st_crs(), [4](#)

sf::st_crs(25830)\$wkt, [3](#)

sfc, [2](#)

sfg, [2](#)

SpatExtent, [2](#)

SpatRaster, [2](#)

SpatVector, [2](#)

terra::crs(), [3](#), [4](#)

terra::ext(), [4](#)

terra::mask(), [4](#)

terra::plot(), [4](#)

terra::plotRGB(), [4](#)

terra::rast(), [3](#), [4](#)

terra::RGB(), [4](#)

terra::vect(), [4](#)

tidyterra::autoplot.SpatRaster(), [4](#)

tidyterra::geom_spatraster_rgb(), [4](#)

tidyterra::pull_crs(), [3](#)