Package 'tiler'

February 20, 2021

Version 0.2.5

Title Create Geographic and Non-Geographic Map Tiles

Description Creates geographic map tiles from geospatial map files or non-geographic map tiles from simple image files.
This package provides a tile generator function for creating map tile sets for use with packages such as 'leaflet'.
In addition to generating map tiles based on a common raster layer source, it also handles the non-geographic edge case, producing map tiles from arbitrary images.
These map tiles, which have a non-geographic, simple coordinate reference system (CRS), can also be used with 'leaflet' when applying the simple CRS option.
Map tiles can be created from an input file with any of the following extensions: tif, grd and nc for spatial maps and png, jpg and bmp for basic images.
This package requires 'Python' and the 'gdal' library for 'Python'.
'Windows' users are recommended to install 'OSGeo4W' (<https://trac.osgeo.org/osgeo4w/>) as an easy way to obtain the required 'gdal' support for 'Python'.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

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

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

SystemRequirements Python (>= 2.7), python-gdal library (For Windows, gdal installed via OSGeo4W <https://trac.osgeo.org/osgeo4w/> recommended) clipboard

Suggests testthat, knitr, rmarkdown, covr, jpeg, bmp, leaflet

Imports sp, rgdal, raster, png

VignetteBuilder knitr

RoxygenNote 7.0.0

NeedsCompilation no

Author Matthew Leonawicz [aut, cre] (<https://orcid.org/0000-0001-9452-2771>)

Maintainer Matthew Leonawicz <mfleonawicz@gmail.com>

Repository CRAN

Date/Publication 2021-02-20 22:20:02 UTC

R topics documented:

`tile` *Create map tiles*

Description

Create geographic and non-geographic map tiles from a file.

Usage

```
tile(
  file,
  tiles,
  zoom,
  crs = NULL,
  resume = FALSE,
  viewer = TRUE,
  georef = TRUE,
  ...
)
```

Arguments

`file`	character, input file.
`tiles`	character, output directory for generated tiles.
`zoom`	character, zoom levels. Example format: `"3-7"`. See details.
`crs`	character, Proj4 string. Use this to force set the CRS of a loaded raster object from `file` in cases where the CRS is missing but known, to avoid defaulting to non-geographic tiling.
`resume`	logical, only generate missing tiles.

viewer	logical, also create preview.html adjacent to tiles directory for previewing tiles in the browser using Leaflet.
georef	logical, for non-geographic tiles only. If viewer = TRUE, then the Leaflet widget in preview.html will add map markers with coordinate labels on mouse click to assist with georeferencing of non-geographic tiles.
...	additional arguments for projected maps: reprojection method or any arguments to raster::RGB, e.g. col and colNA. See details. Other additional arguments lng and lat can also be passed to the tile previewer. See tile_viewer for details.

Details

This function supports both geographic and non-geographic tile generation. When file is a simple image file such as png, tile generates non-geographic, simple CRS tiles. Files that can be loaded by the raster package yield geographic tiles as long as file has projection information. If the raster object's proj4 string is NA, it falls back on non-geographic tile generation and a warning is thrown.

Choice of appropriate zoom levels for non-geographic image files may depend on the size of the image. A zoom value may be partially ignored for image files under certain conditions. For instance using the example map.png below, when passing strictly zoom = n where n is less than 3, this still generates tiles for zoom n up through 3.

Supported file types: Supported simple CRS/non-geographic image file types include png, jpg and bmp. For projected map data, supported file types include three types readable by the raster package: grd, tif, and nc (requires ncdf4). Other currently unsupported file types passed to file throw an error.

Raster file inputs: If a map file loadable by raster is a single-layer raster object, tile coloring is applied. To override default coloring of data and noData pixels, pass the additional arguments col and colNA to .... Multi-layer raster objects are rejected with an error message. The only exception is a three- or four-band raster, which is assumed to represent red, green, blue and alpha channels, respectively. In this case, processing will continue but coloring arguments are ignored as unnecessary.

Prior to tiling, a geographically-projected raster layer is reprojected to EPSG:4326 only if it has some other projection. The only reprojection argument available through ... is method, which can be "bilinear" (default) or"ngb". If complete control over reprojection is required, this should be done prior to passing the rasterized file to the tile function. Then no reprojection is performed by tile. When file consists of RGB or RGBA bands, method is ignored if provided and reprojection uses nearest neighbor.

It is recommended to avoid using a projected 4-band RGBA raster file. However, the alpha channel appears to be ignored anyway. gdal2tiles gives an internal warning. Instead, create your RGBA raster file in unprojected form and it should pass through to gdal2tiles without any issues. Three-band RGB raster files are unaffected by reprojection.

Tiles and Leaflet: gdal2tiles generates TMS tiles. If expecting XYZ, for example when using with Leaflet, you can change the end of the URL to your hosted tiles from {z}/{x}/{y}.png to

`{z}/{x}/{-y}.png`.

This function is supported by two different versions of `gdal2tiles`. There is the standard version, which generates geospatial tiles in TMS format. The other version of `gdal2tiles` handles basic image files like a matrix of rows and columns, using a simple Cartesian coordinate system based on pixel dimensions of the image file. See the Leaflet JS library and `leaflet` package documentation for working with custom tiles in Leaflet.

Value

nothing is returned but tiles are written to disk.

See Also

`view_tiles`, `tile_viewer`

Examples

```
# non-geographic/simple CRS
x <- system.file("maps/map.png", package = "tiler")
tiles <- file.path(tempdir(), "tiles")
tile(x, tiles, "2-3")

# projected map
x <- system.file("maps/map_wgs84.tif", package = "tiler")
tile(x, tiles, 0)
```

---

tiler *tiler: Create map tiles from R*

Description

The tiler package creates geographic map tiles from geospatial map files or non-geographic map tiles from simple image files.

Details

This package provides a tile generator function for creating map tile sets for use with packages such as `leaflet`. In addition to generating map tiles based on a common raster layer source, it also handles the non-geographic edge case, producing map tiles from arbitrary images. These map tiles, which have a non-geographic simple coordinate reference system (CRS), can also be used with `leaflet` when applying the simple CRS option.

Map tiles can be created from an input file with any of the following extensions: tif, grd and nc for spatial maps and png, jpg and bmp for basic images.

This package requires Python and the gdal library for Python. Windows users are recommended to install OSGeo4W: https://trac.osgeo.org/osgeo4w/ as an easy way to obtain the required gdal support for Python in Windows.

---

tiler_options *Options*

Description

Options for tiler package.

Usage

```
tiler_options(...)
```

Arguments

...	a list of options.

Details

On Windows systems, if the system paths for python.exe and OSGeo4W.bat are not added to the system PATH variable, they must be provided by the user after loading the package. It is recommended to add these to the system path so they do not need to be specified for every R session.

As long as you are using OSGeo4W, you can ignore the Python path specification and do not even need to install it on your system separately; OSGeo4W will use its own built-in version.

The recommended way to have GDAL available to Python in Windows is to install OSGeo4W. This is commonly installed along with QGIS.

By default, tiler_options is set on package load with osgeo4w = "OSGeo4W.bat". It is expected that the user has added the path to this file to the system PATH variable in Windows. For example, if it is installed to C:/OSGeo4W64/OSGeo4W.bat, add C:/OSGeo4W64 to your PATH. If you do want to specify the path in the R session using tiler_options, provide the full path including the filename. See the example.

None of this applies to other systems. As long as the system requirements, Python and GDAL, are installed, then tile should generate tiles without getting or setting any tiler_options.

Value

The function prints all set options if called with no arguments. When setting options, nothing is returned.

Examples

```
tiler_options()
tiler_options(osgeo4w = "C:/OSGeo4W64/OSGeo4W.bat")
```

`tile_viewer` *Create an HTML tile preview*

Description

Create an HTML file that displays a tile preview using Leaflet.

Usage

```
tile_viewer(tiles, zoom, width = NULL, height = NULL, georef = TRUE, ...)
```

Arguments

`tiles`	character, directory where tiles are stored.
`zoom`	character, zoom levels full range. Example format: `"3-7"`.
`width`	`NULL` (default) for geospatial map tiles. The original image width in pixels for non-geographic, simple CRS tiles.
`height`	`NULL` (default) for geospatial map tiles. The original image height in pixels for non-geographic, simple CRS tiles.
`georef`	logical, for non-geographic tiles only. If `viewer = TRUE`, then the Leaflet widget in `preview.html` will add map markers with coordinate labels on mouse click to assist with georeferencing of non-geographic tiles.
`...`	additional optional arguments include `lng` and `lat` for setting the view longitude and latitude. These three arguments only apply to geographic tiles. Viewer centering is `0,0` by default.

Details

This function creates a file `preview.html` adjacent to the `tiles` base directory. When loaded in the browser, this file displays map tiles from the adjacent folder. For example, if tiles are stored in `project/tiles`, this function creates `project/preview.html`.

By default, `tile` creates this file. The only reasons to call `tile_viewer` directly after producing map tiles are: (1) if `viewer = FALSE` was set in the call to `tile`, (2) if `tile` was called multiple times, e.g., for different batches of zoom levels, and thus the most recent call did not use the full zoom range, or (3) `preview.html` was deleted for some other reason.

If calling this function directly, ensure that the min and max zoom, and original image pixel dimensions if applicable, match the generated tiles. These arguments are passed to `tile_viewer` automatically when called within `tile`, based on the source file provided to `tile`.

Value

nothing is returned, but a file is written to disk.

See Also

`view_tiles`, `tile`

Examples

```
tile_viewer(file.path(tempdir(), "tiles"), "3-7") # requires existing tiles
```

`view_tiles` *View map tiles with Leaflet*

Description

View map tiles in the browser using leaflet.

Usage

```
view_tiles(tiles)
```

Arguments

`tiles`	character, directory where tiles are stored.

Details

This function opens `preview.html` in a web browser. This file displays map tiles in a Leaflet widget. The file is created when `tile` is called to generate the map tiles, unless `viewer = FALSE`. Alternatively, it is created (or re-created) subsequent to tile creation using `tile_viewer`.

Value

nothing is returned, but the default browser is launched.

See Also

`tile_viewer`, `tile`

Examples

```
# launches browser; requires an existing tile set
## Not run: view_tiles(file.path(tempdir(), "tiles"))
```

Index