Package: gdalio (via r-universe)

August 31, 2024

Title Read Data to via GDAL Warper to an Assumed Grid

Version 0.0.1.9014

Description Convenience wrapper for the GDAL raster warper library.
Set a default context grid, an extent, dimension, projection,
and then read any GDAL raster source into that grid. There are
controls to augment metadata-poor sources (to override missing
or incorrect extent or projection metadata), to set a desired
grid for subsequent use, and ability to control options
available by the GDAL warp library itself. This can be used to
easily read data from any kind of raster data source, local
files, online servers, bare URLs, and database connections.
Data is read in generic form but we provide example wrappers
for commonly used formats for raster data.

License MIT + file LICENSE

Encoding UTF-8

Language es

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.0

LazyData yes

Imports grDevices, memoise, methods, vapour (>= 0.8.0)

URL `https://github.com/hypertidy/gdalio`

BugReports `https://github.com/hypertidy/gdalio/issues`

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

VignetteBuilder knitr

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

RemoteUrl https://github.com/hypertidy/gdalio

RemoteRef HEAD

RemoteSha 5f7827d2460595475cbe0de11ae7a0f3b0b6eae0

Contents

---

gdalio-example-files *Example data files*

Description

Return a path to a data raster file, for easy access to examples.

Usage

```
gdalio_eg_raster()

gdalio_eg_image()
```

Details

This raster has sea surface temperature values in Celsius, with no colour palette defined.

Examples

```
gdalio_eg_raster()
gdalio_eg_image()
```

`gdalio_data`	*Read GDAL raster numeric data*

Description

Data may be one band (the default, first band) or many.

Usage

```
gdalio_graphics(dsn, bands = 1:3, ...)

gdalio_matrix(dsn, ...)

gdalio_array(dsn, ...)

gdalio_image(dsn, ..., extent = NULL, dimension = NULL, projection = NULL)

gdalio_data(dsn, ..., bands = 1L)

gdalio_graphics(dsn, bands = 1:3, ...)

gdalio_matrix(dsn, ...)

gdalio_array(dsn, ...)
```

Arguments

`dsn`	character string, raster source understood by GDAL
`bands`	default 1L, but can be more, duplicated in different order, or 'NULL' for all
`...`	arguments passed to 'vapour::vapour_warp_raster'

Details

`gdalio_data()` returns a list of vectors, `gdalio_matrix()` and `gdalio_array()` and `gdalio_graphics()` return a matrix, array, matrix of the necessary shape, as used by `image()` and `plot()`.

The matrix of hex values returned by `gdalio_graphics()` cannot really be placed on a spatial plot window without a lot of extra work, but it's good for fast visuals to 'plot()' the output. We can write helpers to plot this thing better but WIP atm.

Value

list of numeric vectors

Examples

```
## Not run:
 f <- system.file("extdata/sst.tif", package = "vapour", mustWork = TRUE)

 g <- list(extent = c(130, 160, -60, -30), dimension = c(180, 100),
    projection = "+proj=longlat")
 gdalio_set_default_grid(g)
 v <- gdalio_data(f, band_output_type = "int")
 image(seq(130, 160, length.out = 181), seq(-60, -30, length.out = 101),
    matrix(v[[1]], g$dimension[1])[,g$dimension[2]:1], asp = 1.5)

## End(Not run)
```

`gdalio_data_rgb` *Read GDAL raster data as RGB triples or hex colours*

Description

`gdalio_data_hex` and `gdalio_data_rgb` are a little strange in that they return a vector of character strings and a list of numeric values respectively.

Usage

```
gdalio_data_rgb(dsn, bands = 1:3, ...)

gdalio_data_hex(dsn, bands = 1:3, ...)
```

Arguments

`dsn`	character string, raster source understood by GDAL
`bands`	bands to read, assumes 1:3 (can be 1:4 or any ordering)
`...`	arguments passed to 'vapour::vapour_warp_raster'

Value

'gdalio_data_rgb()' a list of integer vectors, 'gdalio_data_hex()' a character vector of "#" colours

`gdalio_format_source` *Print the code to source format-specific functions*

Description

You can 'source()' the file path displayed by this function to define package-specific formats for the gdalio data.

Usage

```
gdalio_format_source()
```

Details

Running the code in the file path displayed by this function will load functions for terra, stars, raster, and spatstat.

Examples

```
## Not run:
source(gdalio_format_source())

## End(Not run)
```

`gdalio_get_default_grid` *Title*

Description

Title

Usage

```
gdalio_get_default_grid()
```

Value

grid specification (list of extent, dimension, projection)

Examples

```
gdalio_get_default_grid()
gdalio_set_default_grid(list(extent = c(-1000, 1000, -2000, 2000),
 dimension = c(100, 200), projection = "+proj=longlat"))
gdalio_get_default_grid()

gdalio_set_default_grid()
```

gdalio_local_grid *Generate a local grid*

Description

Generate a default local grid to use for subsequent data reads.

Usage

```
gdalio_local_grid(
  x = 147,
  y = -42,
  buffer = 2500000,
  family = "laea",
  dim = if (dev.cur() == 1) {
     c(512, 512)
 } else {
     dev.size("px")
 }
)
```

Arguments

x	longitude
y	latitude
buffer	width either side of x, y
family	projection family (as per PROJ strings)
dim	size of grid nx, ny

Details

All arguments have default values.

Value

list appropriate for `gdalio_set_default_grid()`

Examples

```
gdalio_local_grid()
gdalio_local_grid(family = "stere")
```

`gdalio_set_default_grid` *Title*

Description

Input may be a list with `extent` `$dimension`, `$projection`, which is `c(xmin, xmax, ymin, ymax)`, `c(ncol, nrow)`, and string (accepted by GDAL as a projection input). Alternatively, use a raster, stars, or terra object. Only simple cases of stars will work (regular grids with positive x, negative y transforms).

Usage

```
gdalio_set_default_grid(
  x,
  ...,
  extent = NULL,
  dimension = NULL,
  projection = NULL
)
```

Arguments

`x`	grid specification, a list with '$extent, $dimension, $projection‘ or a spatial grid object see Details

Value

the grid specification (originally: nothing, used to set a default grid available globally)

Examples

```
gdalio_set_default_grid(list(extent = c(-1000, 1000, -2000, 2000),
   dimension = c(100, 200), projection = "+proj=longlat"))
gdalio_set_default_grid()
```

`gdalio_sources` *Data sources for reading*

Description

Provides a data frame of online-accessible data sources. This is very incomplete and has no guarantee of success. It's your responsibility to check usage terms.

Usage

```
gdalio_sources()
```

Details

There is a table of sources 'provider', 'name', 'source' - source is a DSN in GDAL terms, you can query and read from it with GDAL.

Examples

```
srcs <- gdalio_sources()
ttsa <- subset(srcs, provider == "tasmap" & name == "TTSA")$source
ve <- subset(srcs, provider == "gdaltms" & name == "wms_virtualearth_street")$source
gdalio_set_default_grid(list(extent = c(-1, 1, -1, 1) * 800,
                             dimension = rep(min(dev.size("px")), 2L),
                             projection = "+proj=laea +lat_0=-42.8826 +lon_0=147.3257"))
source(gdalio_format_source())
tas_street <- gdalio_terra(ttsa, bands = 1:3, resample = "cubic")
ve_street <- gdalio_terra(ve, bands = 1:3, resample = "cubic")
par(mfrow = c(1, 2))
terra::plotRGB(tas_street)
terra::plotRGB(ve_street)
```

`sources-data` *sources*

Description

sources

`vrt` *Fix missing raster metadata*

Description

Simple metadata augmentation for raster sources.

Usage

```
vrt(x, extent = NULL, projection = NULL)

## Default S3 method:
vrt(x, extent = NULL, projection = NULL)
```

Arguments

`x`	character string, file, url, GDAL dsn
`extent`	numeric 'c(xmin, xmax, ymin, ymax)'
`projection`	character wkt, proj, epsg code

Details

Simple function to add either or both of a raster source extent and projection string.

The attributes from 'extent' as 'source_extent' and/or 'projection' as 'source_projection' are passed directly down to GDAL, via the `gdalio_data()` function, which hands them on to `vapour::vapour_warp_raster()` arguments 'source_extent' and 'source_wkt' respectively.

Value

lightly classed character vector, with "vrt_simple", "character"

Examples

```
vrt("myfile.nc")
vrt("myfile.nc", extent = c(-180, 180, -90, 90))
str(vrt("myfile.nc", extent = c(-180, 180, -90, 90), projection = 4326))
```

Index