Package: lazyraster (via r-universe)

August 30, 2024

Version 0.5.0.9010

Title Generate Raster Data Lazily from 'GDAL'

Description Read raster data at a specified resolution on-demand via 'GDAL' (the Geospatial Data Abstraction Library <https://gdal.org/>). Augments the 'raster' package by never reading data from a raster source until necessary for generating an in-memory 'raster' object. A 'lazyraster' object may be cropped and converted to 'raster' object, and by default will only read a small amount of data sufficient for an overall summary. The amount of data read can be controlled by specifying the output dimensions.

License GPL-3

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 7.1.1

Imports graphics, raster, vapour (>= 0.8.0), methods, quadmesh (>= 0.4.0)

Suggests palr, testthat (>= 2.1.0), covr, knitr, rmarkdown, ggplot2

URL https://github.com/hypertidy/lazyraster

BugReports https://github.com/hypertidy/lazyraster/issues

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

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

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

RemoteRef HEAD

RemoteSha ec99b29ad640f1b78056d7ca330f1e0a67b50591

Contents

`as_raster` *Convert to in-memory raster*

Description

Create an actual raster::raster object by breaking the lazy contract and demanding pixel values at a given resolution.

Usage

```
as_raster(
  x,
  dim = NULL,
  resample = "NearestNeighbour",
  native = FALSE,
  band = 1L
)
```

Arguments

`x`	a lazyraster
`dim`	dimensions, pixel size in rows and columns
`resample`	resampling method, see vapour::vapour_read_raster
`native`	return raster at native resolution, default is `FALSE`
`band`	set to 1-based band numbers to return (independently of those set with `lazyraster()`)

Details

Control the dimensions and method for resampling with the 'dim' and 'resample' arguments.

If `dim` is non-integer it is rounded according to what raster does with the dimensions.

When `native = TRUE` the `dim` argument is ignored, and no resampling is performed.

Value

a regular raster 'BasicRaster' in-memory object

`lazyraster` *Lazy raster*

Description

A lazyraster is a metadata-only shell around a raster file source. Only metadata is read and and used for extent and resolution logic. A lazyraster may be cropped lazily and if plotted or converted to in-memory raster only a reasonable level-of-detail is actually used.

Usage

```
lazyraster(gdalsource, band = NULL, sds = NULL, ...)
```

Arguments

`gdalsource`	a file name or other source string
`band`	which band to use, defaults to all present
`sds`	which subdataset to use, set to 1 if in doubt (see `vapour::vapour_sds_names`)
`...`	ignored for now

Details

See `crop()` for cropping - it works the same but returns a lazyraster, and `as_raster()` for converting to an in-memory raster.

Value

a lazyraster object, a simple shell around a GDAL raster source

Warning

If the inferred Y extents appear to be reversed (ymax > ymin) then they are reversed, with a warning. This occurs for any GDAL data source that does not have a geotransform and so is required for use with raster. This might not be the right interpretation, geotransforms are very general and it might mean the data is meant to be oriented that way. (I don't know why GDAL sets a positive Y pixel height as the default, it's a bit of a pain - should the data be flipped, or should Y be interpreted negative - no way to know!).

Examples

```
sstfile <- system.file("extdata/sst.tif", package = "vapour")
lazyraster(sstfile)
## convert to raster (in memory, but not all of the source)
as_raster(lazyraster(sstfile))
## crop and stay as lazyraster
crop(lazyraster(sstfile), raster::extent(142, 143, -50, -45))
## crop and convert to raster
as_raster(crop(lazyraster(sstfile), raster::extent(142, 143, -50, -45)))
```

lazyraster-methods *Lazy raster S3 methods*

Description

Print and format for lazyraster.

Plot for lazyraster, data pulled on-demand at a reasonable level-of-detail.

Usage

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

## S3 method for class 'lazyraster'
format(x, ...)

## S3 method for class 'lazyraster'
plot(x, y, ...)
```

Arguments

x	a lazyraster
...	passed to raster::plot
y	ignored

Details

Data is pulled from the GDAL source at a resolution suited for the currently open graphics device.

Examples

```
fl <- system.file("images/ga_srtm.png", package = "lazyraster")
print(lazyraster(fl))
## won't work with dumb images with gdalwarp, need rethink
#plot(lazyraster(fl))
```

lazyraster-raster-S4 *Raster methods (S4) for lazyraster.*

Description

These methods are generics from the raster package, extended to work for lazyrasters.

Usage

```
## S4 method for signature 'lazyraster'
crop(x, y, ...)
```

Arguments

x	various (raster, extent)
y	an object with extent (for crop)
...	arguments passed to underlying raster function

Details

For `crop()` this set an active window of data using the same `crop()` function as the raster package. This is the data window that will be pulled by conversion to an actual raster by `as_raster()`.

Value

`lazyraster()` and `crop()` return a lazyraster object, `extent()` returns a regular raster extent

See Also

`raster::raster()`, `raster::extent()`, `raster::crop()`

Examples

```
sstfile <- system.file("extdata/sst.tif", package = "vapour")
lr <- lazyraster(sstfile)
## crop and stay as lazyraster
crop(lazyraster(sstfile), raster::extent(142, 143, -50, -45))
```

---

merc *Mercator extent*

Description

Create an extent in Mercator projection from a longitude, latitude, and a width,height.

Usage

```
merc(x = 146.7, y = -42, wh = 256000)
```

Arguments

x	longitude
y	latitude
wh	width and height, in metres - if only one vaue provided it is also used for height

Value

four values, xmin, xmax, ymin, ymax in global Mercator

Examples

```
merc()  ## a parochial default
library(lazyraster)
vearth <- '<GDAL_WMS> <Service name="VirtualEarth">
<ServerUrl>http://a${server_num}.ortho.tiles.virtualearth.net/tiles/a${quadkey}.jpeg?g=90</ServerUrl></Service
<MaxConnections>4</MaxConnections>    <Cache/>    </GDAL_WMS>'
lr <- lazyraster(vearth)
## Not run: \donttest{
raster::plotRGB(as_raster(crop(lr, merc(-90, 52, 256e4)), band = 1:3))
}
## End(Not run)
```

---

quadmesh *Quadmesh for lazyrasters*

Description

Provides a re-exported generic quadmesh and a method for lazyraster.

Usage

```
## S3 method for class 'lazyraster'
quadmesh(x, z = x, na.rm = FALSE, ..., texture = NULL, texture_filename = NULL)
```

Arguments

x	raster object for mesh structure
z	raster object for height values
na.rm	remove quads where missing values?
...	arguments passed to as_raster, for both x and z if necessary
texture	optional input RGB raster, 3-layers
texture_filename	optional input file path for PNG texture

Details

A quadmesh is a 'rgl::mesh3d' extension, and can be plotted in 3D with 'rgl::shade3d'.

Value

a quadmesh, derived from 'rgl::mesh3d'

Index