Package: ffraster (via r-universe)

August 21, 2024

Title Native Raster Files

Version 0.0.1.9001

Description Provides a loose couplings between the facilities of `raster` and `ff`, to allow a file-backed raster grid to be modified as if it were an array in memory.

Depends R (>= 3.3.0)

Imports configr, ff, raster, yesno

License GPL-3

LazyData true

RoxygenNote 6.1.0

Suggests digest, covr, knitr, rmarkdown, testthat, viridis

VignetteBuilder knitr

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

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

RemoteRef HEAD

RemoteSha d6bd67f992a939e85f2a3d93574b59130440937e

Contents

dim_order *dimension order*

Description

The dimension order as used by ff and raster native grid, see ‘raster::writeRaster‘. The raster package uses the named conventions "BSQ", "BIL" and "BIP" which are domain-specialized ways to record the dimension order of (virtual) 3D arrays. They stand for "band sequential" "band interleaved" and ... correspond to c(1, 2, 3) etc. etc.

Usage

```
dim_order(x, ...)

## S3 method for class 'BasicRaster'
dim_order(x, ...)

## S3 method for class 'character'
dim_order(x, ...)
```

Arguments

x	filename or raster object or ff object
...	ignored

Details

This function always returns an integer vector indicating the order of the dims of a ‘length(dim_order(x))‘ array oriented relative to the R array convention.

Value

integer vector of dimension positions (1-based)

Examples

```
r <- raster::raster(volcano)
dim_order(r)
w <- raster::writeRaster(r, raster::rasterTmpFile(), bandorder = "BIP")
dim_order(w)
dim_order(raster::filename(w))
```

ffrarr *ffraster*

Description

Build a ff array to be used as a raster. For mapping between raster and ff types, see vmode and dataType

Usage

```
ffrarr(dim, mode, filename, readonly = TRUE)
```

Arguments

dim	dimensions in Raster order (nrow, ncol, nlayer)
mode	ff data mode see details
filename	file name as per writeRaster
readonly	open in readonly mode (TRUE is default)

Value

ff

Examples

```
mat <- volcano
library(raster)
b <- brick(raster(mat), raster(mat), raster(mat))
fn <- rasterTmpFile()
dt <- "byte"
ffraster:::.writeGRD(b, dataType = dt, filename = fn)
a <- ffrarr(dim(b), mode = dt, filename = fn, readonly = FALSE)
```

ffraster *ffraster.*

Description

ffraster.

ff_object *Raster and 'ff' file-backed arrays*

Description

Create an file-backed 'ff' object from a raster, or a raster-enabled file.

Usage

```
ff_object(x, readonly = TRUE, filename = NULL, ...)

## S3 method for class 'BasicRaster'
ff_object(x, readonly = TRUE, filename = NULL,
  ...)

## S3 method for class 'character'
ff_object(x, readonly = TRUE, filename = NULL, ...)
```

Arguments

x	raster or raster-able file
readonly	open in read-only mode (TRUE by default)
filename	path to file to create
...	arguments to methods

Details

When the object is created from a filename, 'raster::brick' is used. The 'dim'ension for raster and for ff in this context always keeps degenerate singletons. Please get in touch if this causes you problems.

Value

'ff' object

Examples

```
f <- system.file("extdata", "raster", "sst.grd", package  = "ffraster")
ff_object(raster::brick(f))
if (interactive()) {
  arr <- ff_object(f, filename = "afile.grd")
 }
```

`ff_raster` *Create an ffraster*

Description

Use a raster template to set up an ff array for filling.

Usage

```
ff_raster(x, nlayers = NULL, filename = NULL, ..., setZ = NULL)
```

Arguments

`x`	a raster object, used as the template for the data to come
`nlayers`	the number of layers to instantiate
`filename`	the .grd filename
`...`	arguments passed to .writeGRD
`setZ`	optional z values, must be of length 'nlayers'

Details

Note that no data is transferred, this is a set up function to give an ff array to be populated.

There is currently no control over the bandorder, set to band interleaved (BIL) currently.

Examples

```
## Not run:
n <- 5
files <- raadtools::sstfiles()[1:n, ]

library(raster)
fun <- function(date) raadtools::readsst(date, xylim = extent(120, 160, -50, -30), inputfiles = files)
arr <- ff_raster(fun(files$date[1]), nlayers = n, filename = "afile.grd", overwrite = TRUE)
for (i in seq_along(files$date)) {
arr[,,i] <-   raster::values(t(fun(files$date[i])))
}
# plot(brick("afile.grd"))

## End(Not run)
```

ff_type *ff type from raster type*

Description

ff type from raster type

Usage

```
ff_type(x, ...)

## S3 method for class 'character'
ff_type(x, ...)

## S3 method for class 'BasicRaster'
ff_type(x, ...)

## S3 method for class 'raster_ini'
ff_type(x, ...)
```

Arguments

x	filename, raster object, or ini object
...	ignored

Value

type of ff, see ‘ff::vmode‘

Examples

```
ff_type("LOG1S")
```

ini_file *raster ini file*

Description

Read the configuration file in raw list form, you can use either the filename (.grd) or a raster object.

Usage

```
ini_file(x)

## S3 method for class 'character'
ini_file(x)

## S3 method for class 'BasicRaster'
ini_file(x)
```

Arguments

x	file name or raster

Examples

```
f <- system.file("extdata", "raster", "sst.grd", package  = "ffraster")
ini_file(f)
```

Index