# Package: grout (via r-universe)

August 17, 2024

**Title** Create Raster Tiles

**Version** 0.0.2.9010

**Description** Create raster tiles abstractly.

**Depends** R (>= 3.2.2)

**License** GPL-3

**Suggests** testthat (>= 2.1.0)

**Encoding** UTF-8

**Imports** tibble, vaster (>= 0.0.1.9003)

**URL** https://github.com/hypertidy/grout

**BugReports** https://github.com/hypertidy/grout/issues

**Remotes** hypertidy/vaster

**RoxygenNote** 7.3.1

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

**RemoteUrl** https://github.com/hypertidy/grout

**RemoteRef** HEAD

**RemoteSha** af36753d4436108ff3e5e56f88606fab6e581846

## Contents

grout *Create a tiling scheme from a raster*

## Description

The input may be an actual raster or a matrix. There is an assumed block size of 256x256, and the scheme will record the number of tiles in each dimension and the amount of "overlapping dangle" when the dimensions of the data don't fit neatly within the tiles.

## Usage

```
grout(dimension, extent = NULL, blocksize = NULL, projection = NA_character_)
```

## Arguments

| | |
|---|---|
| dimension | number of columns and rows of the raster grid |
| extent | extent of the raster grid xmin,xmax,ymin,ymax |
| blocksize | tile dimensions in columns (X) and rows (Y) |
| projection | the projection (crs) of the grid |

## Details

If extent is not provided the default 'xmin=0,xmax=ncol,ymin=0,ymax=nrow' is used.

The tile scheme object has print and plot methods for basic information.

See example in the README with 'wk::rct' to generate plot-able and efficient spatial objects from the scheme.

## Value

A "tile scheme" object with information about the tile spacing and extent.

## Examples

```
## one block tile (too big)
grout(c(87, 61), blocksize = c(256L, 256L))
## more size appropriate
grout(c(87, 61), blocksize = c(8, 8))
grout(c(10, 20), c(0, 1, 0, 2), blocksize = c(256, 256))
```

## plot.grout_tiles *plot tiles*

### Description

plot tiles

### Usage

```
## S3 method for class 'grout_tiles'
plot(x, ..., add = FALSE, border = "grey", lwd = 2)
```

### Arguments

| | |
|---|---|
| x | a grout [tiles()] object |
| ... | arguments passed to methods |
| add | add to current plot or start a new one (default is 'FALSE', a new plot) |
| border | the colour of the tile border (default grey) |
| lwd | the width of the tile border (default = 2) |

## print.grout_tiles *print tiles*

### Description

print tiles

### Usage

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

### Arguments

| | |
|---|---|
| x | a grout [tiles()] object |
| ... | ignored |

## tile_index *Tile index*

### Description

Generate a table of tile indexes from a tiling object.

### Usage

```
tile_index(x)
```

### Arguments

| | |
|---|---|
| x | tiling object, created by [tiles()] |

### Details

A data frame with

* **tile** - index, same as raster cell number * **offset_x** - column offset of tile (index column 0-based) * **offset_y** - row offset of tile (index row 0-based, relative to top row) * **ncol** number of columns in tile (the right and bottom margins may have a dangle based on block size) * **nrow** number of rows in th tile * xmin, xmax, ymin, ymax - the exent

Note that ncol,nrow is the block size *unless* the tile is part of a dangling column or row (right or bottom) where the raster doesn't fill complete tiles.

### Value

data frame, see details

### Examples

```
tile_index(grout(c(87, 61), extent = c(0, 1, 0, 1), blocksize = c(32, 16)))
## only one tile in this weird scheme!
tile_index(grout(c(61, 87), blocksize = c(61, 87)))
```

## tile_spec *Create tile specification*

### Description

Take an input grid (dimension and extent) and create a specification of the tiling for that grid within a profile.

## Usage

```
tile_spec(
  dimension,
  extent,
  zoom = 0,
  blocksize = c(256L, 256L),
  profile = c("mercator", "geodetic", "raster"),
  crs = NA,
  xyz = FALSE
)

tile_zoom(
  dimension,
  extent,
  blocksize = c(256L, 256L),
  profile = c("mercator", "geodetic", "raster")
)
```

## Arguments

| | |
|---|---|
| dimension | size in pixels ncol,nrow |
| extent | xmin,xmax,ymin,ymax |
| zoom | the zoom level, starts at 0 and can be up to 24 |
| blocksize | size of each tile, defaults to 256x256 |
| profile | profile domain to use, see Details |
| crs | crs, for raster profile |
| xyz | default is 'FALSE', if 'TRUE' use xyz-mode (zero is at the top) |

## Details

Profiles are 'mercator' or "geodetic" for global systems, or can use "raster" which will use the input grid specification as the entire domain.

'tile_row' in output is in TMS orientation (zero is at the bottom), use 'xyz' arg to switch.

Function 'tile_zoom()' will return the "natural" maximum zoom level, i.e. the zoom at which the tile resolution is just below the input resolution. Note that no reprojection is done, the input extent must match the profile chosen (use 'raster' for native profile).

## Value

data frame with tile specification, tile index, tile_col, tile_row, ncol, nrow, xmin, xmax, ymin, ymax, crs

## Examples

```
tile_spec(c(8194, 8194), c(140, 155, -45, -30), profile = "geodetic")

tile_spec(c(2048, 248), c(140, 155, -45, -30), zoom = 5, profile = "geodetic",
   blocksize = c(512, 512))
```

# Index