Package 'TileManager'

October 4, 2018

Type Package

Title Tile Manager

Version 0.3.0

Date 2018-10-03

Author Andrew Plowright

Maintainer Andrew Plowright <andrew.plowright@alumni.ubc.ca>

Description Tools for creating and detecting tiling schemes for raster data sets.

Depends R (>= 3.5.0)

License GPL (>= 3)

LazyData TRUE

Imports raster, rgeos, sp, utils, methods, graphics, APfun

RoxygenNote 6.0.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-04 04:20:02 UTC

R topics documented:

CHMdemo *Canopy height model demo*

Description

A small section of a canopy height model of a forest in British Columbia, Canada.

Usage

```
CHMdemo
```

Format

A RasterLayer

Cell values are equal to canopy height above ground (in meters)

NonoverlappingBuffers *Non-overlapping buffers*

Description

Takes a set of tiles and removes all overlapping buffer areas, while conserving buffer areas along the edge of the tile set.

Usage

```
NonoverlappingBuffers(buffPolys, unbuffPolys)
```

Arguments

buffPolys	SpatialPolygonsDataFrame. A set of buffered tiles.
unbuffPolys	SpatialPolygonsDataFrame. A set of unbuffered tiles.

Details

When processing a tiled dataset, using buffered tiles can help remove the edge effects along the individual tile borders. However, overlapping buffers generally need to be removed when recombining a series of tiles back into a single raster. Although this can be accomplished by using the unbuffered tile extent, this will also remove the buffered areas along the edge of the tile set. Once these unbuffered tiles are reassembled, the resulting raster will then be smaller than the original dataset before it was tiled.

This may not be a desirable result. The NonoverlappingBuffers function will produce a set of polygons that correspond to the tile extents that conserve buffers only where they do not overlap onto neighboring tiles (i.e.: along the edge of the tile set). These polygons are useful for cropping out overlapping areas from buffered tiles in order to reassemble the tiles into a single raster.

Typically, both inputs for this function would be generated by the `TileScheme` function.

Both `buffPolys` and `unbuffPolys` should represent the same set of tiles. Both require 'col' and 'row' data columns indicating each cell's column and row number.

Value

A SpatialPolygonsDataFrame corresponding to the tiles input in `buffPolys` and `unbuffPolys` with all overlapping buffer areas removed.

`TileDetector` *Tile Detector*

Description

Function for detecting existing tiling scheme from a list of RasterLayers.

Usage

```
TileDetector(inRasters, reord = FALSE)
```

Arguments

`inRasters`	a list of RasterLayers (see raster)
`reord`	logical. If set to `FALSE`, tiles will be stored in the same order in which they appear in `inRasters`. If `TRUE`, tiles will be reordered by column and then by row.

Value

The output of this function is a list of three SpatialPolygonsDataFrame objects:

`tilePolygons`	The tiling grid. Each polygon corresponds to the extent of a single unbuffered tile.
`buffPolygons`	The buffered tiling grid. Each polygon corresponds to the extent of a buffered tile. These polygons overlap with neighboring tiles. If `buffer` is set to 0, this output will be identical to `tilePolygons`.
`nbuffPolygons`	Non-overlapping buffered tiles. These polygons remove overlapping buffers for adjacent tiles, but preserve buffers for tiles on the edges of the tiling grid. Useful for "reassembling" data that had been originally broken into tiles.

TileScheme *Tile Scheme*

Description

This function aims to provide an all-in-one tool for creating tiling schemes, which includes options for overlapping buffers and methods for describing tile sizes in various ways (i.e. using either distance units or cell numbers).

Usage

```
TileScheme(input, dimByCell = NULL, dimByDist = NULL, buffer = 0,
  bufferspill = FALSE, snap = NULL, removeEmpty = FALSE)
```

Arguments

input	filename (character), extent, raster or a vector of four numbers
dimByCell	vector of two numbers. Defines the 'x' and 'y' dimensions of each tile in number of cells. Can only be used when input is a raster.
dimByDist	vector of two numbers. Defines the 'x' and 'y' dimensions of each tile in the distance unit of input (usually meters)
buffer	numeric. If set to >0, overlapping buffers will be created around each tile. Defined in cell number or distance unit, depending on the the usage of dimByCell or dimByDist respectively
bufferspill	logical. Default is FALSE, in which case the tiling grid will be pushed inwards so that the buffers of the outer tiles are within the extent of input. If set to TRUE, the buffers will extend outside of the extent of input
snap	optional. Vector of two numbers corresponding to a pair of coordinates to which the tiling scheme will be aligned. Can only be used in conjunction with dimByDist. The coordinates do not need to b within the extent of input
removeEmpty	logical. Default is FALSE. If set to TRUE, tiles containing only NA cell values will be removed from the tiling scheme. Can only be used when input is a Raster object

Value

The output of this function is a list of three SpatialPolygonsDataFrame objects:

tilePolygons	The tiling grid. Each polygon corresponds to the extent of a single unbuffered tile.
buffPolygons	The buffered tiling grid. Each polygon corresponds to the extent of a buffered tile. These polygons overlap with neighboring tiles. If buffer is set to 0, this output will be identical to tilePolygons.
nbuffPolygons	Non-overlapping buffered tiles. These polygons remove overlapping buffers for adjacent tiles, but preserve buffers for tiles on the edges of the tiling grid. Useful for "reassembling" data that had been originally broken into tiles.

Examples

```
# Create an irregularly shaped grid defined by the number of raster cells
ts1 <- TileScheme(CHMdemo, dimByCell = c(100,120))

# Create an square shaped grid defined by unit distance (m)
ts2 <- TileScheme(CHMdemo, dimByDist = c(50,50))

# Create a grid with buffered cells
ts3 <- TileScheme(CHMdemo, dimByDist = c(50,50), buffer = 5)
```

Index