Package ‘focusmapr’

July 7, 2014

Type Package

Title computation of Focus Maps from a set of input raster or vector layers

Version 1.0

Date 2014-06-11

Author Massimiliano Pittore - GFZ-Potsdam

Maintainer `<pittore@gfz-potsdam.de>`

Description the package implements the concept of Focus Map, a raster description of the combined spatial importance of a set of input layers. The Focus Map can be interpreted as a spatial distribution of data sampling probability given the input layers and the weights of their combination. The Focus Map is generated by a ``normalization'' (or ``mapping'') and a ``pooling'' operation. The package has been implemented in the framework of the FP7-SPACE Project SENSUM (Grant agreement no: 312972).

License GPL

Depends rgdal (>= 0.8.16), raster (>= 2.2.31), spatstat (>= 1.34.0)

R topics documented:

focusmapr-package *Computation of Focus Maps*

Description

The package implements the concept of Focus Maps, a raster description of the spatial data sampling probability (or "importance") as a result of a combination of several raster or vector input layers. Every input layer is "mapped" (usually normalized) and "pooled". Currently two normalizations are available, min-max and logarithmic. In both cases rejection bounds can be specified in order to apply the normalization only with a certain quantile range.

Details

Package:	focusmapr
Type:	Package
Version:	1.0
Date:	2014-06-11
License:	GPL

Author(s)

Massimiliano Pittore

Examples

```
# loading two raster layers from disk
r1 <- system.file("extdata/p1dens.tif", package="focusmapr")
r2 <- system.file("extdata/p2dens.tif", package="focusmapr")
layers <- LoadRasterLayers(c(r1,r2),resamp=TRUE)
# normalize the layers
norm_layers <- lapply(layers,NormalizeLayer)
# create the focus map with a multiplicative pooling and inequal weights
focmap <- FocusMap(norm_layers,pooling="loglinear",weigths=c(.7,.3))
```

FocusMap *create a new* `raster` *object representing a Focus Map from a list of input layers*

Description

the function applies a pooling method to the list of input rasters, returning the result in form of a raster with same extent and resolution. The pooling methods are specified by the argument `pooling`

Usage

```
FocusMap(layers, pooling = "loglinear", weigths)
```

Arguments

`layers`	A list of `raster` objects with same resolution
`pooling`	The desired pooling method. Possible values are \itemlinearadditive pooling \itemloglinearmultiplicative pooling
`weigths`	a vector of (real numbers) weights. The lenght of the vector is equal to the number of input layers. Weights sum to one for the linear pooling, and are not constrained in case of loglinear pooling.

Value

a raster with same extent of the intersection of input layers representing the estimated focus map

Author(s)

Massimiliano Pittore

Examples

```
# loading two raster layers from disk
r1 <- system.file("extdata/p1dens.tif", package="focusmapr")
r2 <- system.file("extdata/p2dens.tif", package="focusmapr")
layers <- LoadRasterLayers(c(r1,r2),resamp=TRUE)
# normalize the layers
norm_layers <- lapply(layers,NormalizeLayer)

# create the focus map with a multiplicative pooling and inequal weights
focmap1 <- FocusMap(norm_layers,pooling="loglinear",weigths=c(.7,.3))

# create another focus map with a additive pooling and equal weights
focmap2 <- FocusMap(norm_layers,pooling="linear",weigths=c(.5,.5))
```

`LoadRasterLayers` *bulk load a set of raster layers from file*

Description

Given a list of filenames (including path) the function loads the different raster layers, returning a list or object of type raster (from package raster). If needed, the function performs flippng, resample and reprojection of the individual layer to obtain an harmonized list.

Usage

```
LoadRasterLayers(layers_paths, repro = FALSE, resamp = FALSE, flip = FALSE)
```

Arguments

`layers_paths`	list of filenames (including complete path) for the layers to be loaded. Supported file types are the 'native' raster package format and those that can be read via rgdal http://cran.at.r-project.org/web/packages/raster/raster.pdf#Rfn.readGDALrgdal.
`repro`	logical, if YES reprojects the layer into a default WGS84 reference system
`resamp`	logical, if YES resamples all the raster layers following the resolution of the first layer. ResampleLayers.
`flip`	logical, if YES a flipping of the input layer around the 'y' axis is performed.The flip options is handy to fix the occasional flipping of rasters created with QGis.

Value

list of `raster` objects (raster).

Author(s)

Massimiliano Pittore

Examples

```
# reads two raster files from a folder

# N.B.: For your own files, omit the system.file and package="focusmapr" bits
# these are just to get the path to files installed with the package

r1 <- system.file("extdata/p1dens.tif", package="focusmapr")
r2 <- system.file("extdata/p2dens.tif", package="focusmapr")
layers<-LoadRasterLayers(c(r1,r2),resamp=TRUE)
```

`MosaicRasters` *Create a tile mosaic of input rasters*

Description

Builds a single mosaic from a set of raster tiles. It uses the `do.Mosaic` function of the package `raster` (raster)

Usage

```
MosaicRasters(paths)
```

Arguments

`paths`	a list of input rasters

Value

a single mosaic raster

Author(s)

Massimiliano Pittore

`NormalizeLayer` *Performs a mapping of input layer through normalization*

Description

Normalizes a single input raster. Normalization either simply re-scales the raster values to fit the interval [0,1], or uses a rejection bounds specified by the argument `rej`. If rejection bounds are used, normalization is applied only within the percentiles defined by the rejection bounds. Rejection bounds are useful to avoid bias from outliers.

Usage

```
NormalizeLayer(rast, norm_type = "linear", rej = NULL)
```

Arguments

`rast`	Input `raster` layer
`norm_type`	Type of normalization. Currently only `norm_type="linear"` is implemented.
`rej`	Rejection bounds. Specify two percentiles outside which normalization is not applied. if `rej=c(0,1)` is applied, this is equivalent to simple normalization.

Value

Normalized `raster` layer

Author(s)

Massimiliano Pittore

Examples

```
# loading two raster layers from disk
r1 <- system.file("extdata/p1dens.tif", package="focusmapr")
r2 <- system.file("extdata/p2dens.tif", package="focusmapr")
layers <- LoadRasterLayers(c(r1,r2),resamp=TRUE)
# normalize the layers with a 0.01 rejection bound
norm_layers<-lapply(layers,FUN=function(x) NormalizeLayer(x,rej=c(0.01,0.99)))
```

`NormalizeLayer_log` *Performs a normalization of input layer through logarithmic transformation*

Description

Normalizes a single input raster. Normalization maps the raster values using a logarithmic mapping to the interval [0,1]. If present, it uses a rejection bounds specified by the argument `rej`. If rejection bounds are used, normalization is applied only within the percentiles defined by the rejection bounds. Rejection bounds are useful to avoid bias from outliers. The logarithmic transformation uses two parameters according to the formula `y=beta0+beta1*log(x)`. If not specified, these parameters are computed automatically.

Usage

```
NormalizeLayer_log(rast, rej = NULL, beta0=NULL, beta1=NULL)
```

Arguments

`rast`	Input `raster` layer
`norm_type`	Type of normalization. Currently only `norm_type="linear"` is implemented.
`rej`	Rejection bounds. Specify two percentiles outside which normalization is not applied. if `rej=c(0,1)` is applied, this is equivalent to simple normalization.
`beta0`	Intercept of the linear model `y=beta0+beta1*log(x)` specifying the logarithmic mapping.
`beta1`	Coefficient of the linear model `y=beta0+beta1*log(x)` specifying the logarithmic mapping.

Value

Normalized `raster` layer

Author(s)

Massimiliano Pittore

Examples

```
# loading two raster layers from disk
r1 <- system.file("extdata/p1dens.tif", package="focusmapr")
r2 <- system.file("extdata/p2dens.tif", package="focusmapr")
layers <- LoadRasterLayers(c(r1,r2),resamp=TRUE)
# normalize the layers with a 0.01 rejection bound
norm_layers<-lapply(layers,FUN=function(x) NormalizeLayer_log(x,rej=c(0.01,0.99)))
```

`ResampleLayers` *Resample the input layers (i.e. changes resolution)*

Description

the function resamples all layers in the list except the reference, which is specified by the `ref_index` argument. The procedure changes the resolution of the processed layers.

Usage

```
ResampleLayers(rasters, ref_index = NULL)
```

Arguments

`rasters`	a list of `raster` objects
`ref_index`	index of the raster layer to be used as reference for resampling. By default=1 (first element of the list)

Value

a list of layers with updated resolution

Author(s)

Massimiliano Pittore

Examples

```
r1 <- system.file("extdata/p1dens.tif", package="focusmapr")
r2 <- system.file("extdata/p2dens.tif", package="focusmapr")
layers <- LoadRasterLayers(c(r1,r2),resamp=TRUE)
# use the resolution of the second layer for resampling the first
resamp_layers<-ResampleLayers(layers,ref_index=2)
```

`SamplingPoints` *Generates a set of sampling points according to a density distribution*

Description

A set of spatial points is generated using the input raster as spatial density distribution. The function creates an inhomogeneous Poisson Point Process to generate the points. A scaling coefficient is used to control the final amount of sampling points.

Usage

```
SamplingPoints(rast, coef)
```

Arguments

`rast`	input raster defining the spatial density of probability of sampling. A FocusMap is typically used
`coef`	scaling coefficient to tune the number of generated points

Value

A set of sampling points.

Author(s)

Massimiliano Pittore

`Vec2Raster` *Convert a vector (shapefile) layer into a raster layer*

Usage

```
Vec2Raster(vec, ras_attr, res)
```

Arguments

`vec`	spatial object of type `SpatialPolygonsDataFrame` with a set of numeric attributes (at least one)
`ras_attr`	attribute of the vector layer used to generate the raster. The attribute must exist, and be a number
`res`	resolution of the output raster. This argument must be provided. The resolution must be specified in the same unit of measure related to the input CRS.

Details

The drivers available will depend on the installation of GDAL/OGR, and can vary; the ogrDrivers() function shows which are available, and which may be written (but all are assumed to be readable). Note that stray files in data source directories (such as *.dbf) may lead to spurious errors that accompanying *.shp are missing.

Value

a raster with same extent as input vector, and resolution specified by the user.

Author(s)

Massimiliano Pittore

Index