SENSUM

Framework to integrate Space-based and in-situ sENSing for dynamic vUlnerability and recovery Monitoring

FP7-SPACE-2012-1 Collaborative Project **312972**

Deliverable

Deliverable	Software Package SW3.2
D3.4	

Workpackage	WP3	**Status (F=Final, D=Draft)**	D
File name			
Dissemination Level (PU=Public; RE=Restricted; CO=Confidential)		PU	

Document Control Page

Version	Date	Comments
1	15.4	First Draft Documentation

Authors	
Name	**Institution**
M. Pittore	GFZ

Deliverable Leader	**Name**	M. Pittore
	Institution	GFZ
Keywords		

Table of Contents

Table of Contents

Illustration Index

List of Acronyms

GFZ	German Research Centre for Geosciences, DE
EUCENTRE	European Centre for Training and Research in Earthquake Engineering, IT
DLR	German Aerospatial Agency, DE
NGI	Norwegian Geotechnical Institute, NO
UCAM	University of Cambridge, UK
CAIAG	Central Asian Institute for Applied Geosciences, KG
IGEES	Institute for Geology and Earthquake Engineering, TJ
ICAT	ImageCat Ltd., UK
EC	European Commission
WP	Workpackage
DoW	Description of Work
ESA	European Space Agency
GIS	Geographic Information System
SQL	Structured Query Language
GEM	Global Earthquake Model
ID	Identifier
OID	Object Identifier
VHR	Very High Resolution
HR	High Resolution
MR	Medium Resolution
OGC	Open Geospatial Consortium
WMS	Web Map Service

WFS	Web Feature Service
WCS	Web Coverage Service
GML	Geographic Markup Language
ISO	International Organization for Standardization
SDI	Spatial Data Infrastructure

Executive Summary

Focus Maps have been designed to provide the end-users with simple tools to understand where to focus the effort of collecting data in the field. A focus map is a representation of the spatial “relevancy” with respect to the set of available information, and constraints. (See SENSUM Deliverable 3.3).

Combining the spatial hazard and the spatial exposure, for instance, a SENSUM user could obtain a focus map defining the spatial density of probability of sampling a location given the level of hazard and the extent of the exposure, therefore increasing the efficiency of the data collection and integration task.

Several algorithms to generate Focus Maps have been implemented in a software package which can easily be accessed by Quantum GIS (QGIS), a popular and FOSS (Free, Open Source Software) GIS environment.

In the following, the QGIS software package and its components are described and and several application examples are provided. The underlying R package is described in the annex I.

1. Introduction

The **qgis-focusmaps tools** are a collection of Quantum GIS (QGIS) scripts and models developed within the FP-7 SENSUM Project to implement algorithms for generating Focus Maps, a key step in the development of efficient data collection schemes.
In QGIS, the *Processing Toolbox*, a geoprocessing environment that can be used to call native and third-party algorithms, is provided. The tools are subdivided in scripts and models, each referring to a specific scripting type.

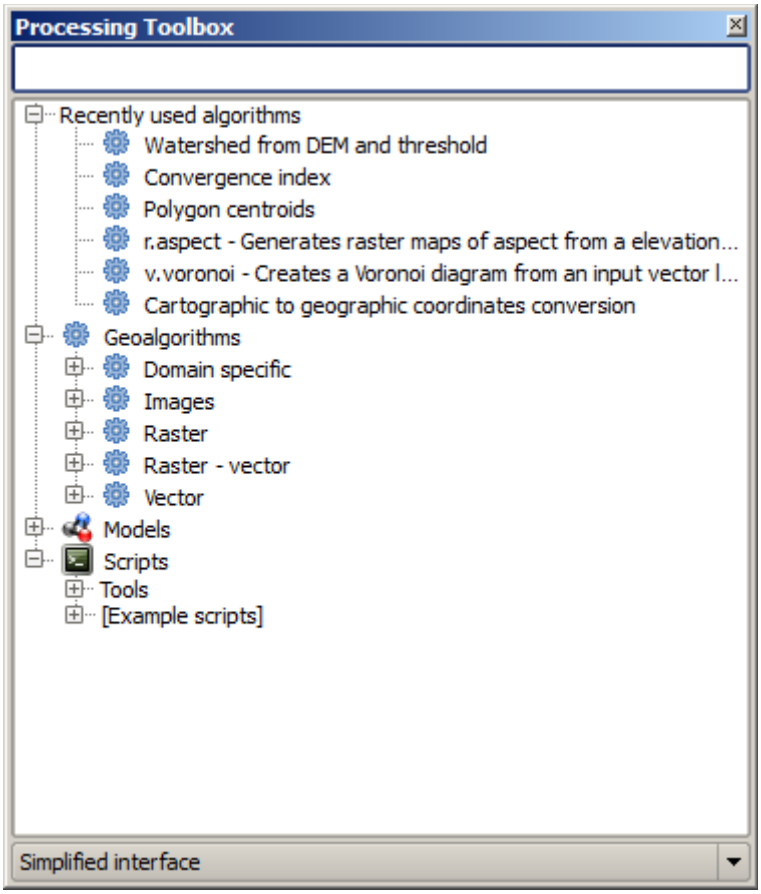

Figure 1: The QGIS Processing Toolbox, showing several available scripts

The *scripts* (shown in Fig. 1) are used to execute a single algorithm or run a batch process based on that algorithm. Each script is a ASCII file containing header and body, where the header contains a set of special instructions to automatically generate the graphical user interface of the script itself. The body of the script in this case is composed by R code. The scripts can acts as simple wrappers for the R packages and function, or can implement original algorithms.
The *models* (an example is provided in Fig. 2) refer to a graphical processing environment recently introduced in QGIS. This framework allows for combining

different algorithms, possibly developed in different environments (R, GRASS, SAGA, OTB, etc) in a single processing pipeline which is defined by visual blocks. We refer to the QGIS online documentation for further details.

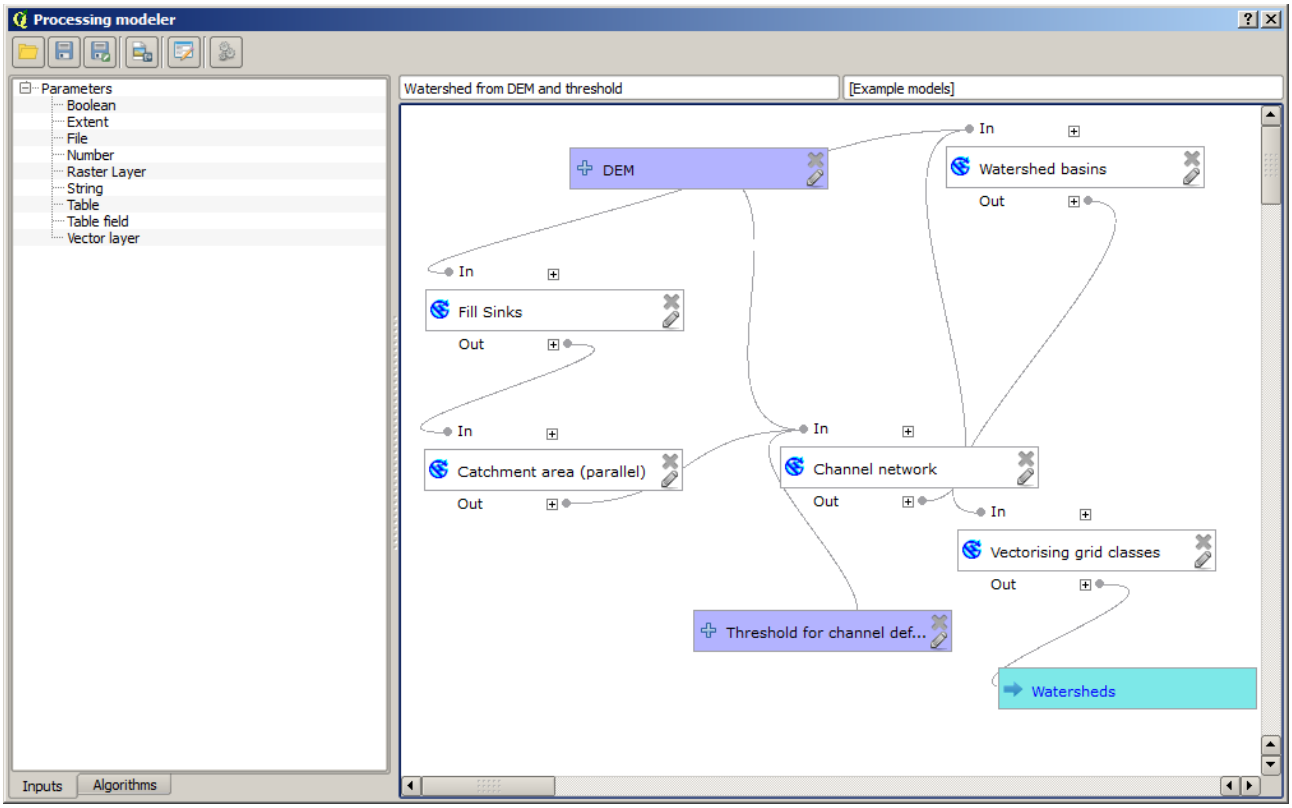

Figure 2: The graphical modeler. Several algorithms can be combined graphically using the modeler to define a workflow, creating a single process that involves several subprocesses.

Contrary to the scripts, the models are saved in a binary format and cannot be directly edited.
The Focus Map utilities are based on the SENSUM R package 'focusmapr'. The Documentation of this package is provided in Annex I.
In the following a brief description of the tools and models is provided.

2. Installation

The scripts and models can be downloaded from https://www.github.com/SENSUM-Project/Qgis-R where the latest development version is residing.
Linux: copy all the files from the folder “rscripts” to the folder “~/.qgis2/processing/rscripts”. Copy all the files from the folder “models” to the folder “~/.qgis2/processing/models”. The files with extension “.help” contain a short documentation about the individual scripts/models.
Windows: same procedure, with the related paths (always in the user´s Documents folder).
The processing scripts are available in QGIS under the Processing->Toolbox menu item, which provides a simple graphical interface. When starting QGIS, all available scripts and models are automatically loaded.
In order to run the scripts, a clean R installation (<= 3.0) has to be present in the system. Please note that models and R scripts have to be explicitly activated in the Processing Toolbox Settings of QGIS.

3. Scripts

GenerateDensity

The script generates a two-dimensional density from a set of points (vector layer), using a kernel density approach. The sigma of the distribution determines the degree of smoothing (hence the granularity) of the distribution. The resolution of the output raster is user-defined.
Note: the script does not accept MultiPoint vectors.

Inputs

vec_layer		Input vector layer (points) containing a set of points
sigma		Parameter of the kernel used to compute the density
resolution_x		X resolution of output raster
resolution_y		Y resolution of output raster

Outputs

dens	raster	Output raster density

GenerateFocusMap

The scripts takes as input a list of raster layers, and combines them to generate a Focus Map. Up to four input rasters can be used. Two different pooling can be chosen, either “linear” or “loglinear”. The normalization is based on a rejection bound approach for outliers rejection. The lower and higher rejection bounds represent the percentile -in the range 0:1 within which the normalization is performed (a rejection bound [0,1] is equivalent to standard normalization). In case the input layers have different resolution and origin, they can be re-sampled (in this case the first element of the list is used as reference).

Note that the current version of the script can accept up to 4 input layers, but the underlying R script does not have such limitations. Moreover, in the R “focusmapr” package several other normalization algorithms are provided.

Inputs

input_rasters	Multiple rasters	List of input raster layers, with same projection (CRS).
high_rejection_bound	number	Higher rejection bound (hrb). 0<=lrb<hrb<=1
low_rejection_bound	number	Lower rejection bound (lrb). 0<=lrb<hrb<=1
pooling_str	string	“linear” (additive) or “loglinear” (multiplicative)
resample_rasters	boolean	If “True” input rasters are resampled according to the first layer´s resolution
equal_weighting	boolean	If “True” weights are set equal to 1/n_input_layers, otherwise the user-provided weights are chosen
weight_1	number	Weight of first raster layer
weight_2	number	Weight of first raster layer
weight_3	number	Weight of first raster layer
weight_4	number	Weight of first raster layer

Outputs

focus_map	Raster	Output raster layer. The extent is the intersection of the input layers´ extents. The resolution is either the native one of

		the input layers, or in case of resampling the one of the first layer

GenerateFocusMap_2inputs

Same functionalities as GenerateFocusMap script, but with two fixed input raster layers, in order to allow for graphical scripting in QGIS using the processing modeller. The input layers must have the same projection (CRS)

Inputs

input_raster1	*raster*	First input raster layer
input_raster2	*raster*	First input raster layer
high_rejection_bound	number	Higher rejection bound (hrb). 0<=lrb<hrb<=1
low_rejection_bound	number	Lower rejection bound (lrb). 0<=lrb<hrb<=1
pooling_str	string	“linear” (additive) or “loglinear” (multiplicative)
resample_rasters	boolean	If “True” input rasters are resampled according to the first layer´s resolution
equal_weighting	boolean	If “True” weights are set equal to 1/n_input_layers
weight_1	number	Weight of first raster layer
weight_2	number	Weight of first raster layer

Outputs

focus_map	raster	Output raster layer. The extent is the intersection of the input layers´ extents

RasterFlip_y

Performs a flipping of the input raster around 'y' axis.

NOTE: This script can be used when, for instance, a density distribution from a set of points (heatmap) is generated using the QGIS “Heatmap” plugin. In this

case the resulting raster is interpreted by the GDAL loader of the SENSUM tools as flipped and must be corrected.

Inputs

raster_layer	raster	Input raster layer

Outputs

output_layer	raster	Output raster layer

Models

Models are processing pipelines which graphically combine one or several QGIS scripts. Different types of scripts can be mixed (R, GRASS, SAGA, OTB, POSTGIS, etc) in a single workflow with defined inputs and outputs. The result is itself a script. The models can be used to implement specific sequences of operations (from preprocessing stages to complete processing batches) for later reuse. The use of a graphical composition makes the development of models rather intuitive. In the section “Models” at page 13 an example of the creation of a new model is provided.

Demo_cologne_1

The script generates a set of sampling points based on a focus map. The focus map is generated from two layers, the extent of a flood scenario and the density of buildings (see Fig. 3). This model has been designed to implement a simple demo. The input files to run the demo are available for download in the github repository of the software package.

The generated points are automatically written in a table of a PostgreSQL/PostGIS database.

NOTE In order to generate the raster building density distribution used in the current example, the QGIS heatmap plugin or the SENSUM GenerateDensity script can both be used. In case the QGIS heatmap plugin is used, the *RasterFlip_y* script have to be used to correct the orientation of the raster.

The density is combined with the expected flood scenario to generate a loglinear focus map. The resulting focus map is then used to generate a

distribution of sampling points, through the *GenerateSamplingPoints* script.

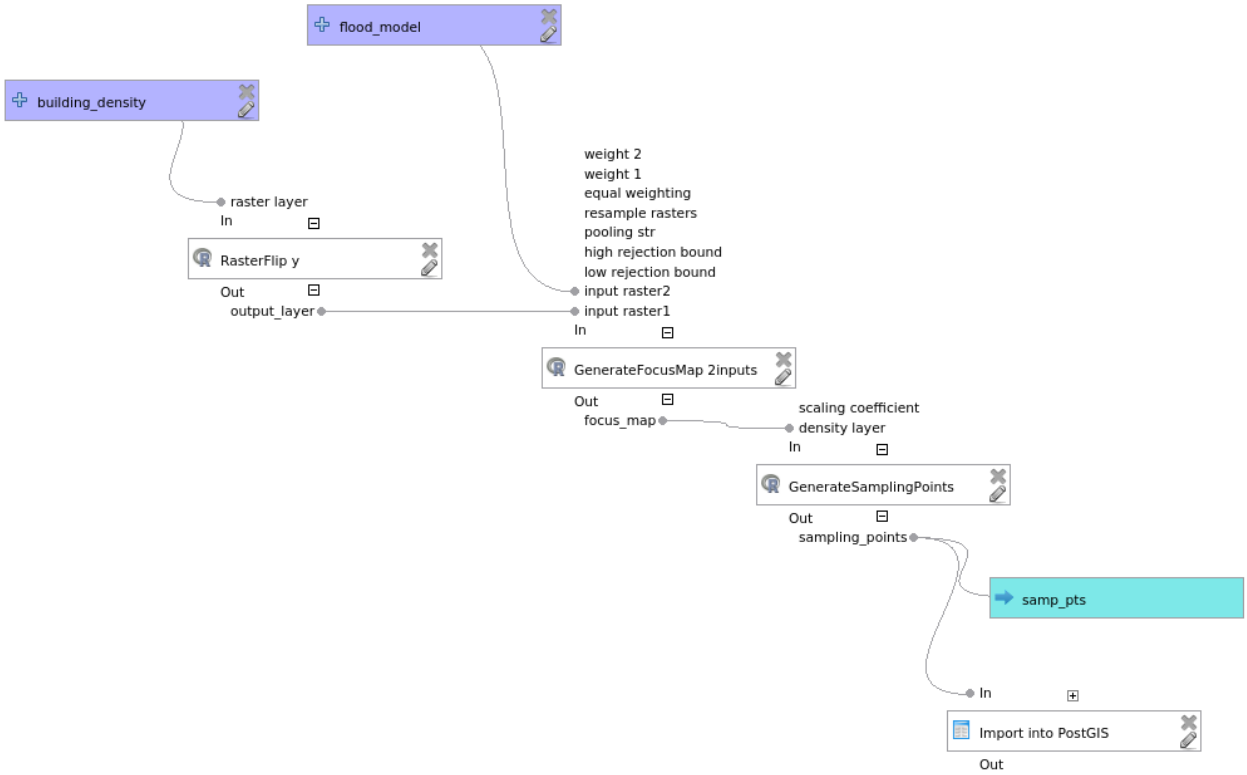

Figure 3: Structure of the QGIS model demo_cologne_1. In purple the input rasters, in cyan the output vector. The white blocks represent processing stages

Inputs

building_density	raster	Density of buildings, generated using the QGIS heatmap plugin. The density raster has been generated with the QGIS heatmap utility, and must be flipped in order to be further processed.
flood_model	raster	Flooding scenario

Outputs

samp_pts	vector	Set of sampling points generated according to the focus map

Example

In the following an example of usage of the above described software packages is shown.
NOTE: The data necessary to run the example can be downloaded from the public repository https://github.com/SENSUM-project/Qgis-R.

Generating a focus map

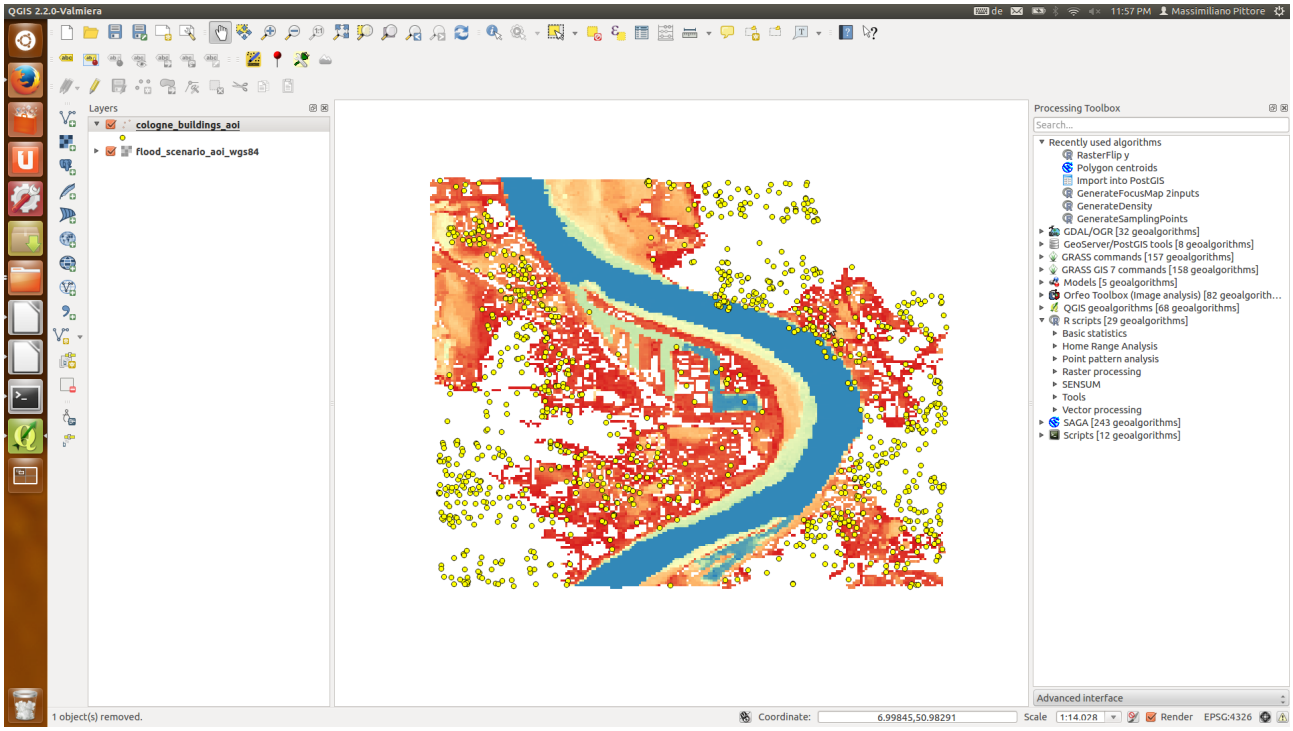

Figure 4 Cologne test case: flooding scenario and residential buildings´ distribution

In Fig. 4 the input data are shown. In this example, the Cologne (Germany) test case is considered. The available data is composed by a flooding scenario and the spatial distribution of residential buildings in the area possibly exposed to flooding.
Our goal is to generate a focus map which combines the hazard (flooding) layer with the exposure (buildings) layer.
As first operation, a raster describing the density of building is generated by using the *GenerateDensity* script. As input layer we choose the vector (shapefile) containing the buildings´ centroids in the area (see Fig. 16). The kernel sigma is chosen (0.03 degrees, since the layer´s projection is WGS84) such to find a tradeoff between the granularity and the informativeness of the resulting density.
Several values of the parameters can be used, and the most significant result

can be later picked up.
In Fig. 6 the resulting density is shown, with superimposed the building centroids.

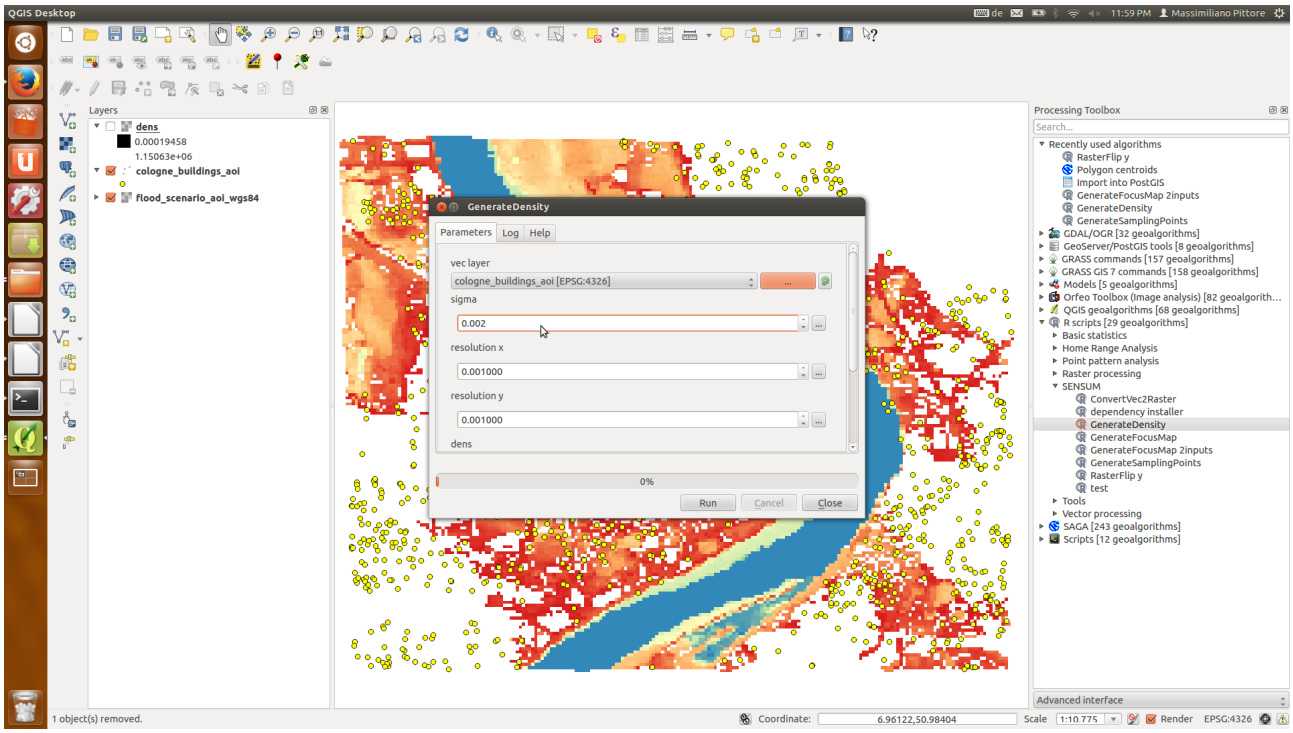

Figure 5 Generation of an heatmap density map of the residential buildings.

It is now possible to generate a focus map by combining the building density layer and the hazard layer. The script *GenerateFocusMap* is therefore selected

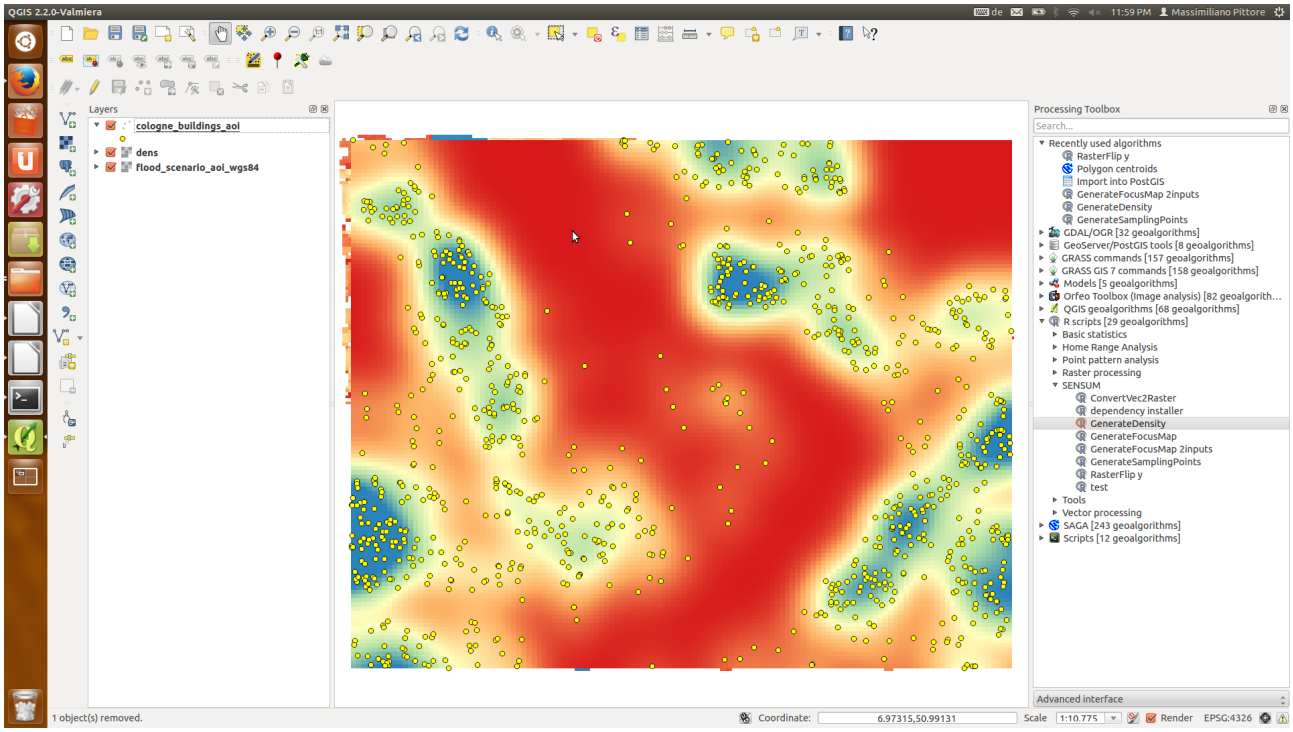

Figure 6 Resulting kernel density of the buildings´ distribution

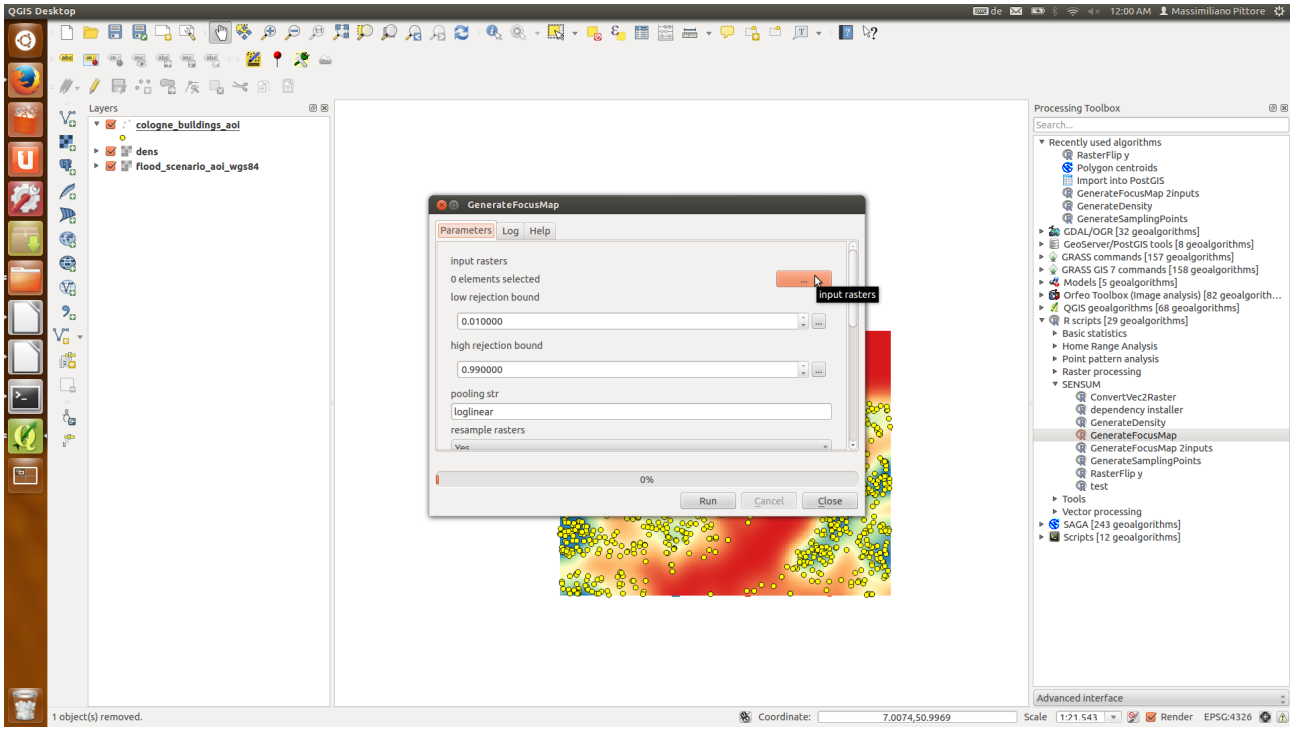

Figure 7 Interface of the GenerateFocusMap *script, part of the SENSUM scripts in the left processing menu*

from the SENSUM processing tools (on the right of the QGIS environment, in

Fig. 7.
The input rasters are selected by clicking on the input button. The layers are selected from the working environment (see Fig. 8).
In this example we leave unchanged all of the options of the script. Since by default the equal weighting scheme is active, the script assigns a 0.5 weight to each of the raster layers.
The chosen pooling operator is loglinear, as default.
Once the parameters are correctly set, by clicking on '*run*' the script is processed.
The resulting focus map is shown in Fig. 9 As we can observe, the areas which are both exposed to flooding, and shows higher building density are highlighted by the focus map.

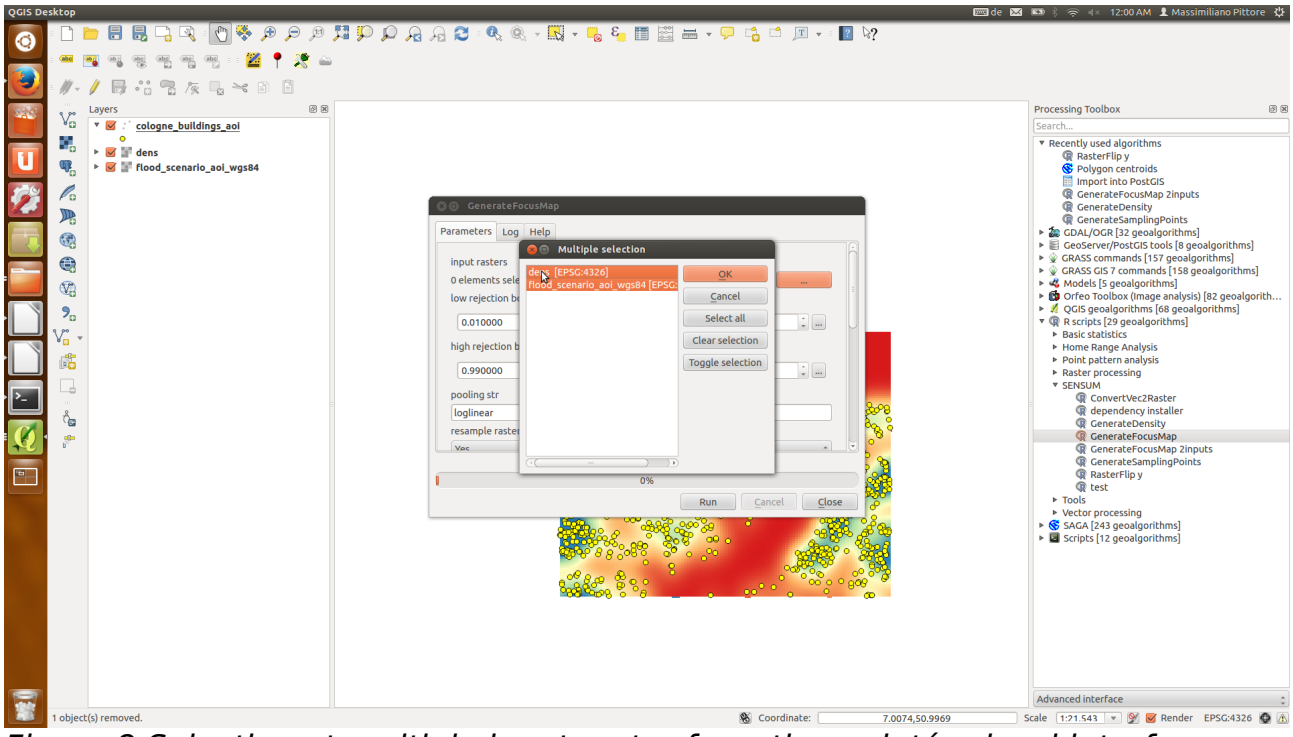

Figure 8 Selection ot multiple input raster from the script´s visual interface

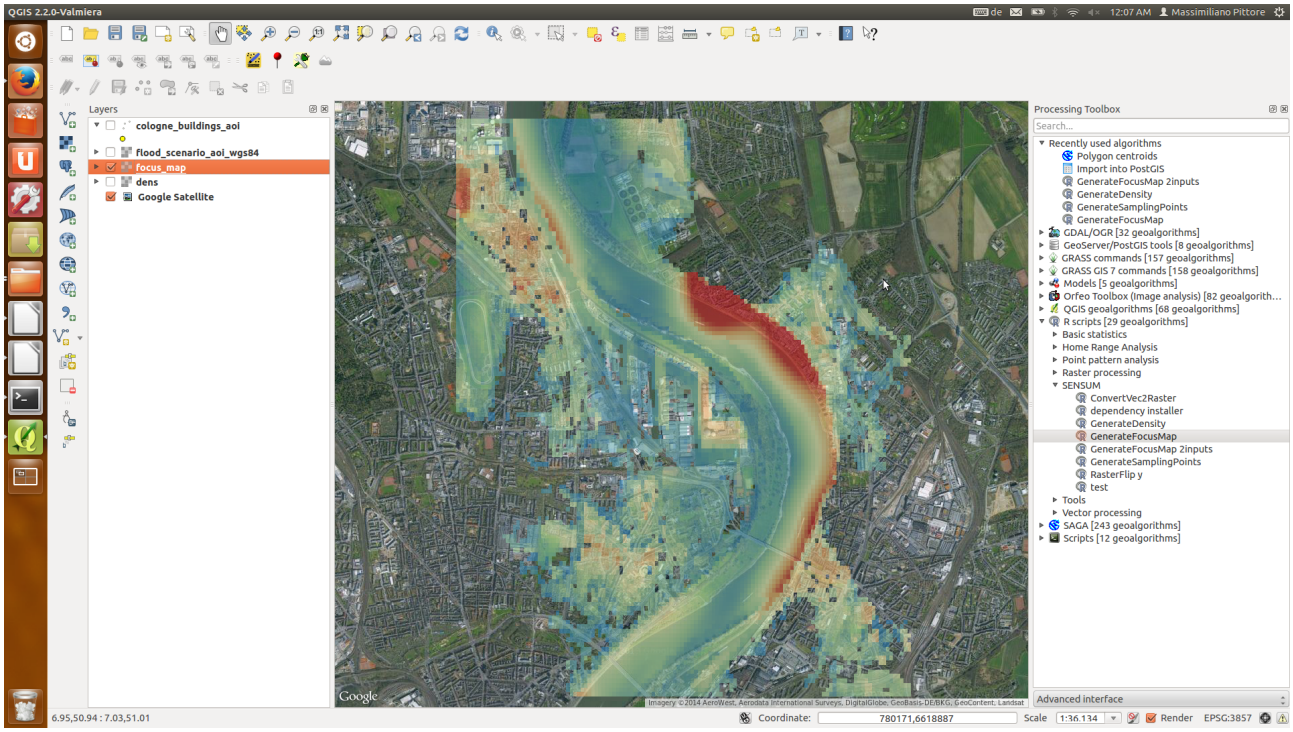

Figure 9 Resulting focus map. In background, the city of Cologne is visualized through the openlayers plugin.

Annex I – the focusmapr R package

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