Package 'defore Stable'

May 18, 2021

Type Package

Title Classify jpeg images into forest or not forest using the color intensities of red, green and blue.

Version 0.1.0

Date 2021-05-12

Author Jesper Muren, Dmitry Otryakhin

Maintainer Jesper Muren <jespermuren@gmail.com>

Description Implements two out-of box classifiers designed by Muren and Otryakhin [2021] for distinguishing forest and non-forest terrain images. Under these algorithms, there are frequentist approaches: one parametric, using stable disitributions and the Cramér-von Mises test, and another one- non parametric, using the squared Mahalanobis distance. The package also contains functions for data handling and building of new classifiers as well as some test data set.

License GPL-3

Encoding UTF-8

Depends stabledist,
R (>= 4.0.5)

Imports goftest,
jpeg,
plyr,
raster,
StableEstim

RoxygenNote 7.1.1

R topics documented:

`Forest_Tester_AD` *Anderson darling tester with argument for parameter.*

Description

Anderson darling tester with argument for parameter.

Usage

```
Forest_Tester_AD(params, dataset)
```

Arguments

`params`	data.frame with stable distribution parameters
`dataset`	data.frame a data frame with colour intensities corresponding to red, green and blue channels of the input image.

Value

a data frame with statistics of AD tests corresponding to red, green and blue channels of the input image.

Examples

```
library(deforeStable)
library(raster)
data("geoimages")

obj <- geoimages[[2]]
plotRGB(obj, scale=1, asp=1)

mtrx <- as.matrix(obj)
pars <- Koutparams(mtrx)

dd_ad<-Forest_Tester_AD(params=pars, dataset=mtrx)
```

`Forest_Tester_CVM` *Cramer von Mises tester with argument for parameter*

Description

Performs Cramer-von Mises goodness-of-fit tests of empirical distributions of colour intensities in the three channels against pre-specified stable distributions. It is designed to be used in conjunction with Koutparams, and argument params is a data frame returned by the latter. dataset is a data frame with colour intensities of the image. It has the same format as the one produced by read_data: observations are in rows and each of the 3 columns correspond to red, green or blue colour.

Usage

```
Forest_Tester_CVM(params, dataset)
```

Arguments

params	data.frame with stable distribution parameters
dataset	data.frame a data frame with colour intensities corresponding to red, green and blue channels of the input image.

Value

a data frame with statistics of CvM tests corresponding to red, green and blue channels of the input image.

Examples

```
library(deforeStable)
library(raster)
data("geoimages")

obj <- geoimages[[2]]
plotRGB(obj, scale=1, asp=1)

mtrx <- as.matrix(obj)
pars <- Koutparams(mtrx)

dd_cvm <- Forest_Tester_CVM(params=pars, dataset=mtrx)
dd_cvm
```

---

geoimages *RGB images of forests, cities and farms.*

Description

A list containing aeral images of 3 types of places: forests, farmlands and cities. All of them are from Scandinavia. There are 27 forest, 15 city and 7 farmland pictures. A table, describing the images are saved in a table named geoimages_desc.

Usage

```
geoimages
```

Format

A list of 49 objects of an S4 class RasterStack containing RGB 0-1 images. Every element of the list has colour intensities wrapped into variable

layers 1 is for intensity of red, 2- green, 3- of blue channel

Source

QGIS google plugin

Koutparams *Koutrouvelis parameter estimation of image data*

Description

In data, there are three columns and each column corresponds to the color intensity of one channel: red, green and blue correspondingly. The four parameters: alpha, beta, gamma and delta, of the stable distribution is estimated for each of these channels using the Koutrouvelis regressions-type technique.

Usage

```
Koutparams(data)
```

Arguments

data	matrix or data frame with color intensities of red, green and blue for an image.

Value

a data frame with columns alpha, beta, gamma, delta and rows red, green and blue.

Examples

```
library(deforeStable)
library(raster)
data("geoimages")

obj <- geoimages[[2]]
plotRGB(obj, scale=1, asp=1)

mtrx <- as.matrix(obj)
pars <- Koutparams(mtrx)
pars
```

mahalaclust *Clusters images of forest*

Description

Clusters images of forest using hierarchical clustering and the squared Mahalanobis distance in order to reduce the number of parameter sets used for testing in the parametric model implemented in Param_classifier. Images that are clustered together are assumed to be from the same distribution and have their estimated stable distribution parameters averaged and returned. Inputs are a list of forest image data on matrix form, the number of clusters to cluster the data into and a list of data frames containing the estimated stable distribution parameters for the provided forest images.

Usage

```
mahalaclust(ForestData, clusters, Forest_params)
```

Arguments

ForestData	list of named matrices containing data of forest image color intensities, as returned by read_data.
clusters	a numeric giving the number of clusters to divide the forest data into
Forest_params	list of named data frames with stable distribution parameters, as retuned by function Koutparams.

Value

list of data frames with averaged stable distribution parameter set for each cluster.

Examples

```
library(deforeStable)
library(raster)
data("geoimages")
data("geoimages_desc")

data <- geoimages[1:5]

datalist <- lapply(data, as.matrix)
parlist <- lapply(datalist, Koutparams)
names(datalist) <- as.list(geoimages_desc[1:5,1])
names(parlist) <- as.list(geoimages_desc[1:5,1])

clustpars <- mahalaclust(ForestData = datalist, clusters = 2, Forest_params = parlist)
clustpars
```

Mahala_dist *Squared Mahalanobis distance between two samples*

Description

Calculates the squared Mahalanobis distance between two distributions of colour intensities. Argument params is a data frame with the reference disitribution to test against and argument dataset is a data frame with the distribution to test. Both have the same format as the one produced by read_data: observations are in rows and each of the 3 columns correspond to red, green or blue colour.

Usage

```
Mahala_dist(params, dataset)
```

Arguments

params	data.frame with colour intensities corresponding to red, green and blue channels of the input image.
dataset	data.frame with colour intensities corresponding to red, green and blue channels of the input image.

Value

a numeric with value of squared Mahalanobis distance.

Examples

```
library(deforeStable)
library(raster)
data("geoimages")

obj1 <- geoimages[[2]]
obj2 <- geoimages[[11]]
plotRGB(obj1, scale=1, asp=1)
plotRGB(obj2, scale=1, asp=1)

mtrx <- as.matrix(obj1)
pars <- as.matrix(obj2)

dd_Sqmahala <- Mahala_dist(params=pars, dataset=mtrx)
dd_Sqmahala
```

MultipleTester *Do multiple parametric tests and return best one*

Description

Performs multiple hypothesis tests according to function func of empirical data data against a list of parameters params. func and params must be compatible, apart from that there are no specific restrictions imposed on them.

Usage

```
MultipleTester(data, params, func)
```

Arguments

data	a data frame with colour intensities corresponding to red, green and blue channels of the input image.
params	a list of data frames with parameter values
func	a function that performs a test and returns its statistic

Value

data frame with smallest statistic.

Examples

```
library(deforeStable)
library(raster)
data("geoimages")
obj <- geoimages[[10]]
plotRGB(obj, scale=1, asp=1)
```

```
mtrx <- as.matrix(obj)
pars <- Koutparams(mtrx)

mpt <- MultipleTester(data=mtrx, params=list(pars, pars), func=Forest_Tester_CVM)
mpt
```

NonParamCV *Cross-validation for the Non-parametric model using the squared Mahalanobis distance*

Description

Performs k-fold cross-validation to find the optimal threshold for the squared Mahalanobis distance to use for the non parametric model implemented in function Nonparam_classifier. This is done by splitting images into smaller sub-images and and for each sub-image the squared Mahalanobis distance is computed with all forest images in the data set and the smallest distance is chosen. These distances are compared to a range of thresholds and the threshold which produces the best classification result according to the accuracy is returned. As input it takes a path to a directory containing forest images, a path to a directory contain non-forest images, the side length of the sub-images the images are split into and the number of folds to use for the k-fold cross-validation. Computations are parallelized.

Usage

```
NonParamCV(forestdir, Nonforestdir, n_pts = 7, nrfolds = 5)
```

Arguments

forestdir	character string with path to directory containing forest images.
Nonforestdir	character string with path to directory containing non-forest images.
n_pts	numeric for sub-image side length, default=7.
nrfolds	numeric for number of folds in k-fold cross-validation, default=5.

Value

list with first element: numeric with best accuracy and threshold which produced it. Second element: data frame with all tested thresholds and corresponding accuracy and other performance metrics. Third element: List of data frames with color intensities of forest images used to train.

Examples

```
library(deforeStable)
library(raster)
library(doParallel)

forestdir <- "forest image directory path"
Nonforestdir <- "Non-forest image directory path"
NonparCV <- NonParamCV(forestdir = forestdir, Nonforestdir = Nonforestdir,
                       n_pts = 7, nrfolds = 5)
```

```
test_image <- read_data_raster(filename, dir)

Nonpar_pred <- Nonparam_classifier(test_image, n_pts = 7, references = NonparCV[[3]],
                           thres = NonparCV[[1]][[1]])
jpeg::writeJPEG(image=Nonpar_pred, target='Nonpartest_im.jpeg')
```

`Nonparam_classifier` *Classification of an image using the non parametric model based on the squared Mahalanobis distance.*

Description

Predict whether parts of a given image contains the terrains of provided reference images by splitting the given image into smaller sub-images and testing these against reference images. The argument rastData is the image to be classified on the RasterStack format. Argument n_pts the side length of square sub-images the image is split into. Argument references is a list of data frames with colour intensities of reference images. Argument thres is a threshold value for the squared Mahalanobis distance, deciding if a sub-image belongs to reference terrain or not. The reference images and threshold should be obtained from the NonParamCv function. Parallel decides if computations should be parallelized.

Usage

```
Nonparam_classifier(rastData, n_pts, references, thres, parallel = TRUE)
```

Arguments

`rastData`	RasterStack as returned by function read_data_raster.
`n_pts`	numeric for sub-image side length.
`references`	list of data frames with same format as the one produced by function read_data.
`thres`	numeric threshold for squared Mahalanobis distance
`parallel`	logical for activating parallel computing.

Value

A matrix of same dimension as input image, with element values 0 or 1. Element value 0 indicates that pixel does not contain reference terrain, 1 indicates the opposite.

Examples

```
library(deforeStable)
library(raster)
library(doParallel)
data("geoimages")

obj <- geoimages[[28]]
ref <- geoimages[1:5]
```

```
reflist <- lapply(ref, as.matrix)


NonparClass <- Nonparam_classifier(rastData = obj, n_pts = 7, references = reflist, thres = 25, parallel = TRUE)
jpeg::writeJPEG(image = NonparClass, target = 'NonparClass.jpeg')
```

`ParamCV` *Cross-validation for the parametric model using the Cramér-von Mises statistic and stable distributions.*

Description

Performs k-fold cross-validation to find the optimal threshold for the Cramér-von Mises statistic to use for the parametric model implemented in function Param_classifier. This is done by splitting images into smaller sub-images and and for each sub-image the Cramér-von Mises statistics for each of the three color intensities red, green and blue is computed with all forest images in the data set and the smallest sum of statistics is chosen. These statistics are compared to a range of thresholds and the threshold which produces the best classification result according to the accuracy is returned. As input it takes a path to a directory containing forest images, a path to a directory contain non-forest images, the side length of the sub-images the images are split into and the number of folds to use for the k-fold cross-validation. Further, it takes a logical deciding if clustering of parameter sets should be included, to reduce computational time. Lastly, it takes the max threshold for the range checked against when trying to find the optimal Cramér-von Mises threshold. Computations are parallelized.

Usage

```
ParamCV(
  forestdir,
  Nonforestdir,
  n_pts = 7,
  nrfolds = 5,
  clustering = TRUE,
  maxt = 14
)
```

Arguments

`forestdir`	character string with path to directory containing forest images.
`Nonforestdir`	character string with path to directory containing non-forest images.
`n_pts`	numeric for sub-image side length, default=7.
`nrfolds`	numeric for number of folds in k-fold cross-validation, default=5.
`clustering`	logical for deciding if clustering of forest image parameter sets should be included, default=TRUE.
`maxt`	numeric for the max threshold range that the CvM statistic is checked against to maximize accuracy, default=14.

Value

list with first element: numeric with best accuracy and threshold which produced it. Second element: data frame with all tested thresholds and corresponding accuracy and other performance metrics. Third element: List of data frames with parameter sets for the stable distributions of the color intensities of forest images used to train.

Examples

```
library(deforeStable)
library(raster)
library(doParallel)

forestdir <- "forest image directory path"
Nonforestdir <- "Non-forest image directory path"
ParCV <- ParamCV(forestdir = forestdir, Nonforestdir = Nonforestdir,
                   n_pts = 7, nrfolds = 5, clustering = TRUE,
                   maxt = 14)

test_image <- read_data_raster(filename, dir)

Par_pred <- Param_classifier(test_image, n_pts = 7, pars = ParCV[[3]],
                          thresh = ParCV[[1]][1:3])
jpeg::writeJPEG(image=Par_pred, target='Partest_im.jpeg')
```

Param_classifier *Classification of an image using the parametric model based on the Cramér-von Mises statistics*

Description

Predict whether parts of a given image contains the terrains of provided reference stable distributions by splitting the given image into smaller sub-images and testing these against reference disitributions. The argument rastData is the image to be classified on the RasterStack format. Argument n_pts is the side length of square sub-images the image is split into. Argument pars is a list of data frames with parameters for the stable distributions of colour intensities of the reference images. Argument thres is threshold values for the squared Cramér-von Mises statistics, deciding if a sub-image belongs to reference terrain or not. Parameter sets and threshold should be obtained from ParamCV function. Parallel decides if computations should be parallelized.

Usage

```
Param_classifier(rastData, n_pts, pars, thres, parallel = TRUE)
```

Arguments

rastData	RasterStack as returned by function read_data_raster.
n_pts	numeric for sub-image side length.
thres	numeric thresholds for Cramér-von Mises statistics
parallel	logical for activating parallel computing.
references	list of data frames with stable distribution parameters, as produced by function Koutparams.

Value

A matrix of same dimension as input image, with element values 0 or 1. Element value 0 indicates that pixel does not contain reference terrain, 1 indicates the opposite.

Examples

```
library(deforeStable)
library(raster)
library(doParallel)
data("geoimages")

obj <- geoimages[[28]]
ref <- geoimages[1:5]

reflist <- lapply(ref, as.matrix)
parlist <- lapply(reflist, Koutparams)

parClass <- Param_classifier(rastData = obj, n_pts = 7, pars = parlist, thres = c(15,15,15), parallel = TRUE)
jpeg::writeJPEG(image = parClass, target = 'parClass.jpeg')
```

---

read_data *Import a jpeg image*

Description

read_data is to read jpeg images and return a 3-column data.frame with pixels in rows and red, green, blue in columns. read_data_matrix reads jpeg images and return 3 matrices for each of red, green and blue colors. read_data_raster imports jpeg as a raster object.

Usage

```
read_data(filename, dir)

read_data_matrix(filename, dir)

read_data_raster(filename, dir)
```

Arguments

filename	name of the jpeg file to import
dir	the directory where the image is located

Functions

- read_data_matrix: returns three matrices
- read_data_raster: returns a RasterStack object

Examples

```
dd<-read_data(filename, dir)
hist(dd[,1])

dd<-read_data_matrix(filename, dir)
jpeg::writeJPEG(image=dd[[1]], target='ex.jpeg')

dd<-read_data_raster(filename, dir)
plotRGB(dd, scale=1, asp=1)
```

Index