Package 'rasclass'

May 2, 2016

Type Package

Title Supervised Raster Image Classification

Version 0.2.2

Date 2016-04-29

Author Daniel Wiesmann <daniel.wiesmann@tecnico.ulisboa.pt> and David Quinn <djq@urbmet.com>

Maintainer Daniel Wiesmann <daniel.wiesmann@tecnico.ulisboa.pt>

Description Software to perform supervised and pixel based raster image classification. It has been designed to facilitate land-cover analysis. Five classification algorithms can be used: Maximum Likelihood Classification, Multinomial Logistic Regression, Neural Networks, Random Forests and Support Vector Machines. The output includes the classified raster and standard classification accuracy assessment such as the accuracy matrix, the overall accuracy and the kappa coefficient. An option for in-sample verification is available.

License GPL (>= 2)

LazyLoad yes

Imports methods, car, nnet, RSNNS, e1071, randomForest

NeedsCompilation no

Repository CRAN

Date/Publication 2016-05-02 06:31:45

R topics documented:

`rasclass-package` *Supervised Raster Image Classification*

Description

This package is built to perform supervised, per-pixel based raster image classification.

Details

The raster image classification is carried out by calling a sequence of functions that load data, calculate the classification grid and produce an accuracy assessment of the classification.

The package contains the readRasterFolder function to load a set of external data layers from a folder containing raster images in the ESRI asciigrid format (*.asc* file extension). All files in the specified folder are parsed and stored in a rasclass-class object. The requirements for reading the input raster files is that they are all in the same projection, are aligned and have the same grid-size (i.e. all raster files should have the same header). Furthermore one raster file has to be specified as sample data. Alternatively, data can also converted into the rasclass format from a dataframe using the setRasclassData function.

For the classification, the package contains five supervised, per-pixel classification methods: Maximum Likelihood Classification, Multinomial Logistic Regression, Neural Networks, Random Forests and Support Vector Machines. There is only one classification function classifyRasclass and the algorithm can be specified with as an argument. The output of the classifications is the classified raster grid and standard accuracy assessment indicators, including user and producer accuracies, the overall accuracy, the confusion matrix and the kappa coefficient.

Author(s)

Daniel Wiesmann <daniel.wiesmann@ist.utl.pt> and David Quinn <djq@mit.edu>

See Also

`rasclass-class`, `rasclassRaster-class`,

`readRaster`, `writeRaster`,

`readRasterFolder`, `setRasclassData`,

`buildFormula`, `checkRasclass`,

`rasclassMlc`, `classifyRasclass`

Examples

```
## Not run:
# If available, load data from external folder
object <- readRasterFolder(path = "mypath", samplename = "mysample",
filenames = c('myvar1.asc', 'myvar2.asc'))

## End(Not run)
```

```
# For this example, create artificial data
mysample <- c(rep(rep(c(1,2), each = 25), 25), rep(rep(c(3,4), each = 25), 25))
mysample <- mysample + sample(c(0, NA), 2500, replace = TRUE, prob = c(1, 50))
myvar1 <- rep(1:50, each = 50) + rnorm(2500, 0, 5)
myvar2 <- rep(rep(1:50), 50) + rnorm(2500, 0, 5)
newdata <- data.frame(mysample, myvar1, myvar2)

# Prepare a rasclass object using the dataframe and specifying raster properties
object <- new('rasclass')
object <- setRasclassData(newdata, ncols = 50, nrows = 50,
xllcorner = 0, yllcorner = 0, cellsize = 1, NAvalue = -9999,
samplename = 'mysample')

# Classify using each algorithm once
outlist <- list()
outlist[['maximumLikelihood']] <- classifyRasclass(object, method = 'maximumLikelihood')
summary(outlist[['maximumLikelihood']])

outlist[['logit']] <- classifyRasclass(object, method = 'logit')
summary(outlist[['logit']])

outlist[['neuralNetwork']] <- classifyRasclass(object, method = 'neuralNetwork')
summary(outlist[['neuralNetwork']])

outlist[['randomForest']] <- classifyRasclass(object, method = 'randomForest')
summary(outlist[['randomForest']])

outlist[['supportVector']] <- classifyRasclass(object, method = 'supportVector')
summary(outlist[['supportVector']])

# Store sample data as a rasclassRaster for display purposes
mysample.ras <- new('rasclassRaster')
mysample.ras@grid <- mysample
mysample.ras@nrows <- 50
mysample.ras@ncols <- 50
mysample.ras@xllcorner <- 0
mysample.ras@yllcorner <- 0
mysample.ras@cellsize <- 1
mysample.ras@NAvalue <- -9999

# Plot results of each classifier
opar <- par(mfrow = c(2, 3))
image(mysample.ras)
title('Sample data')
for(i in 1:length(outlist)) {
image(outlist[[i]]@predictedGrid)
title(names(outlist)[[i]])
}
par(opar)
```

buildFormula *Build a formula for Raster Classification*

Description

This function builds a formula from the dataframe from the data slot in the specified rasclass object.

Usage

```
buildFormula(object, varlist = NULL)
```

Arguments

object	A rasclass-class object with non-empty data and samplename slots.
varlist	An optional character vector containing the names of the variables that shall be used in the classification algorithms.

Details

A formula is built automatically using all the columns in the dataframe from the data slot of the specified object. The formula is stored in the call slot of the rasclass-class object. The dependent variable in the formula will be the name specified in the samplename slot of the given input object.

If not all columns from the data slot should be used for classifiaction, the list of variables to include can be specified using the optional argument *varlist*. The samplename is specified previously when adding data to the rasclass-class object with the corresponding functions and should not be included in the varlist argument.

Value

A rasclass-class object with the newly built formula in the formula slot.

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

readRaster, writeRaster,

readRasterFolder, setRasclassData,

checkRasclass,

rasclassMlc, classifyRasclass

Examples

```
## Not run:
# Load data from external folder
object <- readRasterFolder(path = "mypath", samplename = "mysample",
filenames = c('myvar1.asc', 'myvar2.asc'))

## End(Not run)

# For this example, create artificial data
mysample <- c(rep(rep(c(1,2), each = 25), 25), rep(rep(c(3,4), each = 25), 25))
mysample <- mysample + sample(c(0, NA), 2500, replace = TRUE, prob = c(1, 10))
myvar1 <- rep(1:50, each = 50) + rnorm(2500, 0, 5)
myvar2 <- rep(rep(1:50), 50) + rnorm(2500, 0, 5)
myvar3 <- sample(1:2500)
newdata <- data.frame(mysample, myvar1, myvar2, myvar3)

# Prepare a rasclass object using the dataframe and specifying raster properties
object <- new('rasclass')
object <- setRasclassData(newdata, ncols = 50, nrows = 50,
xllcorner = 0, yllcorner = 0, cellsize = 1, NAvalue = -9999,
samplename = 'mysample')

# Classify and show results using all columns
object <- classifyRasclass(object)
summary(object)

# Change formula to exclude one variable
object <- buildFormula(object, varlist = c('myvar1', 'myvar3'))

# Classify and show results
object <- classifyRasclass(object)
summary(object)
```

---

checkRasclass *Check rasclass object for internal consistency*

Description

This function checks wether a rasclass-class object is internally consistent.

Usage

```
checkRasclass(object)
```

Arguments

object	A rasclass object.

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

readRaster, writeRaster,

readRasterFolder, setRasclassData,

buildFormula,

rasclassMlc, classifyRasclass

Examples

```
# Instanciate rasclass object
object <- new('rasclass')

# Create artificial data
# For this example, create artificial data
mysample <- c(rep(rep(c(1,2), each = 25), 25), rep(rep(c(3,4), each = 25), 25))
mysample <- mysample + sample(c(0, NA), 2500, replace = TRUE, prob = c(1, 10))
myvar1 <- rep(1:50, each = 50) + rnorm(2500, 0, 5)
myvar2 <- rep(rep(1:50), 50) + rnorm(2500, 0, 5)
newdata <- data.frame(mysample, myvar1, myvar2)

# Prepare a rasclass object using the dataframe and specifying raster properties
object <- new('rasclass')
object <- setRasclassData(newdata, ncols = 50, nrows = 50,
xllcorner = 0, yllcorner = 0, cellsize = 1, NAvalue = -9999,
samplename = 'mysample')

# The object passes the test at this point
checkRasclass(object)

# Manually change some values number of rows to an inconsitent value
object@gridSkeleton@nrows <- 12345
object@samplename <- 'wrongName'

# The rasclass object now fails the test
checkRasclass(object)
```

---

classifyRasclass *Classsifier function for rasclass objects*

Description

Classifies data stored rasclass objects with one of five different classification algorithms. Also predicts the resulting output grid and calculates accuracy measures (accuray matrix, overall accuracy and kappa coefficient).

Usage

```
classifyRasclass(rasclassObj, splitfraction = 1, method = 'logit', ...)
```

Arguments

`rasclassObj`	A `rasclass-class` object containing the data for classification.
`method`	An optional argument to choose the classification algorithm. The default is 'logit', other options are 'maximumLikelihood', 'neuralNetwork', 'randomForest' or 'supportVector'.
`splitfraction`	An optional `numeric` argument specifying a fraction for in-sample verifictaion.
`...`	Optional additional arguments passed on to the classification functions. This can be used to run the classification algorithms with settings that are different from the default.

Details

With this function, data is classified using one out of the following five classification algorithms: Maximum Likelihood Classification, Multinomial Logistic Regression, Neural Networks, Random Forests or Support Vector Machines (see section 'Classification methods' for details of each algorithm). The algorithm can be specified using the `method` argument, with 'logit' as default. Most classification algoritms are imported from other packages. To choose specific settings for the chosen algorithm, additional arguments specified in '...' will be passed on to the classification function.

The optional argument 'splitfraction' can be used for *in-sample verfication*. If 'splitfraction' differs from zero, the data will be randomly split into two parts with the fraction specified. The classification model will then be trained on one fraction and the other fraction will be used for prediction. The accuracy measures are then calculated as an in-sample verification, only comparing accuracy in data that was not used for training the model. The 'training' slot of the rasclass object stores the splitting information as a logical vector.

Details of each classification algortithm are described below.

Value

A `rasclass-class` object, containing the classified raster, the classification object itself and standard accuracy measures: accuracy matrix, user and producer accuracies, overall accuracy and the kappa coefficient. All the outputs are stored in their corresponding slots of the output object.

Classification methods

The classification methods used here are mostly imported from other packages. The description of the details of the classifiers can be found in the documentation files of the imported functions. An algorithm can be chosen using the `method` argument. To pass arguments to the chosen classifier, additional arguments can be given that are passed on to the classifier functions.

Gaussian Maximum Likelihood Classifier
The maximum likelihood classifier is implemented directly in this package. It is a parametric classifier and assumes normal probability distributions in each class. Detailed descriptions of the classifier can be found in the paper by Paola (2005) or in standard textbooks. Specify 'maximumLikelihood' in the methods argument to use this algorithm.

Multinomial Logistic Regression
The logit algorithm is imported from the `multinom` function in the `nnet` package. Details can be found in the documentation of the functions. Specify 'logit' in the methods argument to use this algorithm.

Neural Networks
Neural networks are implemented in the R Stuttgart Neural Network Simulator package. The function used for classification is the mlp function from the RSNNS package. Specify 'neuralNetwork' in the methods argument to use this algorithm.

Random Forests
The Random Forest classifier is imported from the randomForest package using the randomForest function. Specify 'randomForest' in the methods argument to use this algorithm.

Support Vector Machines
Support Vector Machines are implemented in R in the e1071 package, which is an interface for the external C library libsvm. The function used for claassification is svm. Specify 'supportVector' in the methods argument to use this algorithm.

References

Paola, J. D., Schowengerdt, R. A. (1995). A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification. *IEEE Transactions on Geoscience and Remote Sensing*, 33(4), 981-996.

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

readRaster, writeRaster,

readRasterFolder, setRasclassData,

buildFormula, checkRasclass,

rasclassMlc

Examples

```
## Not run:
# If available, load data from external folder
object <- readRasterFolder(path = "mypath", samplename = "mysample",
filenames = c('myvar1.asc', 'myvar2.asc'))

## End(Not run)

# For this example, create artificial data
mysample <- c(rep(rep(c(1,2), each = 25), 25), rep(rep(c(3,4), each = 25), 25))
mysample <- mysample + sample(c(0, NA), 2500, replace = TRUE, prob = c(1, 50))
myvar1 <- rep(1:50, each = 50) + rnorm(2500, 0, 5)
myvar2 <- rep(rep(1:50), 50) + rnorm(2500, 0, 5)
newdata <- data.frame(mysample, myvar1, myvar2)

# Prepare a rasclass object using the dataframe and specifying raster properties
object <- new('rasclass')
object <- setRasclassData(newdata, ncols = 50, nrows = 50,
xllcorner = 0, yllcorner = 0, cellsize = 1, NAvalue = -9999,
samplename = 'mysample')

# Classify using each algorithm once
```

```
outlist <- list()
outlist[['maximumLikelihood']] <- classifyRasclass(object, method = 'maximumLikelihood')
summary(outlist[['maximumLikelihood']])

outlist[['logit']] <- classifyRasclass(object, method = 'logit')
summary(outlist[['logit']])

outlist[['neuralNetwork']] <- classifyRasclass(object, method = 'neuralNetwork')
summary(outlist[['neuralNetwork']])

outlist[['randomForest']] <- classifyRasclass(object, method = 'randomForest')
summary(outlist[['randomForest']])

outlist[['supportVector']] <- classifyRasclass(object, method = 'supportVector')
summary(outlist[['supportVector']])

# Store sample data as a rasclassRaster for display purposes
mysample.ras <- new('rasclassRaster')
mysample.ras@grid <- mysample
mysample.ras@nrows <- 50
mysample.ras@ncols <- 50
mysample.ras@xllcorner <- 0
mysample.ras@yllcorner <- 0
mysample.ras@cellsize <- 1
mysample.ras@NAvalue <- -9999

# Plot results of each classifier
opar <- par(mfrow = c(2, 3))
image(mysample.ras)
title('Sample data')
for(i in 1:length(outlist)) {
image(outlist[[i]]@predictedGrid)
title(names(outlist)[[i]])
}
par(opar)
```

`rasclass-class` *Class "rasclass"*

Description

This class object is a container for all the variables used in the classification algorithms of the `rasclass` package. The methods provided in the package sequentially fill the slots with data and results.

Objects from the Class

Objects can be created by calls of the form `new("rasclass")`.

Slots

`path`: Object of class `character`. The path from which the data is loaded.

`data`: Object of class `data.frame`. The dataframe that contains the data from the loaded ascii files.

`samplename`: Object of class `character`. The name of a column in the `data` slot that will be used as sample in the supervised classification.

`formula`: Object of class `formula`, storing the formula passed on to the classification algorithm.

`call`: Object of class `call`, storing the last classification call applied to the object.

`gridSkeleton`: Object of class `rasclassRaster`, containing the skeleton of the output grid. In the structure of rasclass-class objects, only values that are different from null in every input layer (except the sample) are considered in the analysis.

`training`: Object of class `logical`. Is used if the the argument 'splitfraction' of the 'classifyRasclass' function has been used to split data for an in-sample verification. the vector used to split the data is stored in this slot.

`maximumLikelihood`: Object of class `list`. This list is created when the Maximum Likelihood Classifiaction is preformed on the rasclass-class object. It containins the mean vectors and inverse covariance matrices for each class.

`randomForest`: Object of class `randomForest`. This is created when using the Random Forest classification.

`logit`: Object of class `multinom`. Created when using the Random Forest classification.

`neuralNetwork`: Object of class, `mlp`. Created when using the Neural Network classification.

`supportVector`: Object of class `svm`. Created when using the Support Vector Machines classification.

`predictedGrid`: Object of class `rasclassRaster`. The predicted grid resulting from the classification.

`overallAccuracy`: Object of class `numeric`. This slot is used to store the overall accuracy of the classification.

`accuracyMatrix`: Object of class `matrix`. The accuracy matrix of the classification, including user and producer accuracies.

`kappa`: Object of class `numeric`. The kappa accuracy coefficient of the classification, a measure for the accuracy of the classification.

Methods

buildFormula Builds a formula for classification, based on column names of the data in rasclass object.

checkRasclass Checks internal consistency of rasclass object.

classifyRasclass Classification using one of five algorithms.

readRasterFolder Loads data from several asciigrid files in a folder.

setRasclassData Sets data in rasclass object based on a dataframe.

summary Prints a summary of the rasclass-object.

image Plots the classified raster from the rasclass-object.

View Shows the data frame in the rasclass-object.

See Also

rasclass-package, rasclassRaster-class,
readRaster, writeRaster,
readRasterFolder, setRasclassData,
buildFormula, checkRasclass,
rasclassMlc, classifyRasclass

`rasclassMlc` *Maximum likelihood classifier*

Description

This function is the rasclass implementation of the Maximum Likelihood Classifier.

Usage

```
rasclassMlc(rasclassObj)
```

Arguments

`rasclassObj`	A `rasclass` object.

Details

This function is used in the wrapping function `classifyRasclass` to perform Maximum Likelihood Classification. It is provided for review and possible alterations, but it is not recommended for direct use, since this function does not include accuracy assessment or in-sample verification. Therefore it is recomended to perform Maximum Likelihood Classification through the function `classifyRasclass` using the 'methods' argument.

The Maximum Likelihood Classification assumes that for each class, the probability of class membership can be described by a multidimensional Gaussian probability density function. Under this assumption, the maximum likelihood estimates of the probability density function parameters are simply the mean vector and covariance matrix of the subset of each class of the training data. The probability of class membership for prediction is then determined by these estimated parameters and the classification is done by selecting the most probable class for each observation. For details see Paola & Schowengerdt (2005) or statistical textbooks.

Value

A `rasclass-class` object with the results of the Maximum Likelihood Classification.

References

Paola, J. D., Schowengerdt, R. A. (1995). A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification. *IEEE Transactions on Geoscience and Remote Sensing*, 33(4), 981-996.

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

readRaster, writeRaster,

readRasterFolder, setRasclassData,

buildFormula, checkRasclass,

classifyRasclass

---

rasclassRaster-class *Class "rasclassRaster"*

Description

The rasclass representation of ESRI Ascii grids.

Objects from the Class

Objects can be created by calls of the form `new('rasclassRaster')`.

Slots

`ncols`: The number of columns of the raster grid.

`nrows`: The number of rows of the raster grid.

`xllcorner`: Coordinates of the X coordinate of the lower left corner.

`yllcorner`: Coordinates of the Y coordinate of the lower left corner.

`cellsize`: The cell size of the grid.

`NAvalue`: The value in the raster that represents NA values.

`grid`: A numeric vector containing all grid values. The rows of the grid are sequentially appended to this vector.

Methods

readRaster Reads rasters from the ESRI asciifile format.

writeRaster Writes rasters in the ESRI asciifile format.

image Plots the rasclass-raster.

References

ESRI ASCII raster format definition.

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/ESRI_ASCII_raster_format/009t0000000z000000/

See Also

rasclass-package, rasclass-class,

readRaster, writeRaster,

readRasterFolder, setRasclassData,

buildFormula, checkRasclass,

rasclassMlc, classifyRasclass

---

readRaster *Read ESRI asciigrid files*

Description

This function reads ESRI asciigrid files and stores the information in a rasclassRaster object.

Usage

```
readRaster(path, asInteger = FALSE)
```

Arguments

path	A local path to the input ascii raster file.
asInteger	An optional logical argument. If set TRUE, the input data values will be rounded and stored as integers.

Details

The information from the header of the ESRI asciigrid raster (*.asc* file extention) is stored in specific slots of the rasclassRaster object. The gridcell values are stored in a numeric vector. The rows of the raster grid are sequencially appended to that vector.

The optional argument allows to load data as integer to reduce memory requirements of the rasclassRaster object.

Value

A rasclassRaster object.

References

ESRI ASCII raster format definition.

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/ESRI_ASCII_raster_format/009t0000000z000000/

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

writeRaster,

readRasterFolder, setRasclassData,

buildFormula, checkRasclass,

rasclassMlc, classifyRasclass

Examples

```
## Not run:
myraster <- readRaster(path = "mypath", asInteger = FALSE)
image(myraster)
## End(Not run)
```

---

readRasterFolder *Load ESRI asciigrid Data for Classification*

Description

This function automatically loads all ESRI asciigrid files from a specified folder into a rasclass-class object.

Usage

```
readRasterFolder(path, samplename = "sample", filenames = NULL,
object = new("rasclass"), asInteger = FALSE)
```

Arguments

path	A path to a folder that contains input raster files (*.asc* extention).
samplename	An optional character string containing the name of the sample file, the default name is "sample" or equivalently "sample.asc".
filenames	An optional character vector containing the names of the files used as explanatory (dependent) variables in the classifictaion.
object	An optional rasclass-class object to store the data in.
asInteger	An optional logical variable, whether the data should be loaded as integer values to reduce the memory requirements.

Details

This function loads ESRI asciigrid files (*.asc* file extention) that are found in the specified folder. All files in the folder will be loaded if not specified differently using the *filenames* argument. The data is stored into the `data` slot of a `rasclass-class` object, wich can be used for classification using one of the rasclass algorithms. The names of the files provided are stored in the column names of the stored dataframe.

It is required that all the input raster files in the specified folder have the same extent and gridsize (i.e. have the same header) and are in the same projection. The rasclass classifier methods assume that all rasters are aligned and have he same grid size. An identical header and projection system of all the files assures this comparability of all input layers in the subsequent classification.

The rasclass classifiers are supervised classification algorithms and therefore a sample file has to be provided. The sample file contains the training cells for the models. The default sample file name is "sample", if the sample file has another name it can be specified using the optional argument *samplename*. The ".asc" extension is not required in the filenames, they are added and stripped off, depending on the use of the names.

Value

A `rasclass-class` object containing the loaded data as a dataframe in the `data` slot.

Memory Issues

The `readRasterFolder` function only keeps track of data raster cells that have an observed value in every input layer provided, except for the sample layer. Therefore if any of the layers representing the independent variables in the classification has a NA value in a cell, none of the cell values will be stored. The resulting grid structure of the minimum area with complete inforamtion of each cell is stored as a dummy variable vector in the gridSkeleton slot.

If there are memory constraints when loading large raster files, the order of reading the input raster files can be specified using the filenames option to optimize memory usage. In that way, by specifying the input raster with the most `NA` values as first file to read, the number of cells kept from all the other raster files read subsequently is reduced.

See Also

`rasclass-package`, `rasclass-class`, `rasclassRaster-class`,

`readRaster`, `writeRaster`,

`setRasclassData`,

`buildFormula`, `checkRasclass`,

`rasclassMlc`, `classifyRasclass`

Examples

```
## Not run:
object <- readRasterFolder(path = "mypath", samplename = "mysample",
filenames = c('myvar1.asc', 'myvar2.asc'))
object <- classifyRasclass(object)
## End(Not run)
```

`setRasclassData` *Add data from dataframe to rasclass object*

Description

This function adds data from a dataframe to an existing `rasclass-class` object.

Usage

```
setRasclassData(newdata, object = new('rasclass'), ncols = NA, nrows = NA,
xllcorner = NA, yllcorner = NA, cellsize = NA, NAvalue = NA,
samplename = 'sample')
```

Arguments

`newdata`	A data frame containing raster cell values in its columns.
`object`	An optional rasclass object where the data should be updated to, a new object is created if not specified.
`ncols`	The number of columns of the raster grid. Optional if the values are already specified in the rasclass object.
`nrows`	The number of rows of the raster grid. Optional if the values are already specified in the rasclass object.
`xllcorner`	Coordinates of the X coordinate of the lower left corner. Optional if the values are already specified in the rasclass object.
`yllcorner`	Coordinates of the Y coordinate of the lower left corner. Optional if the values are already specified in the rasclass object.
`cellsize`	The cell size of the grid. Optional if the values are already specified in the rasclass object.
`NAvalue`	The value in the raster that represents NA values. Optional if the values are already specified in the rasclass object.
`samplename`	Optional name of the column in the input data frame that will be used as sample data. The default is 'sample'.

Details

This function adds data from a dataframe to a rasclass object for subsequent classification using the classifiers from the rasclass package. Similar to the `readRasterFolder`, the name of the column containing the sample data has to be specified. The default sample column name is "sample".

The data in the input dataframe is assumed to be from raster files with the same projection, grid size and location. For ascii files this is equivalent to having an identical header in all the layers. If the common header has not been specified in the `gridSkeleton` slot of the input object before, the values have to be given as arguments to the function.

Value

A rasclass-class object containing the loaded data as a dataframe in the data slot.

Memory Issues

The setRasclassData function only keeps track of data raster cells that have an observed value (non-NA) in every column of the input dataframe provided. The resulting grid structure will be retained in the gridSkeleton slot.

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

readRaster, writeRaster,

readRasterFolder,

buildFormula, checkRasclass,

rasclassMlc, classifyRasclass

Examples

```
# For this example, create artificial data
mysample <- c(rep(rep(c(1,2), each = 25), 25), rep(rep(c(3,4), each = 25), 25))
mysample <- mysample + sample(c(0, NA), 2500, replace = TRUE, prob = c(1, 10))
myvar1 <- rep(1:50, each = 50) + rnorm(2500, 0, 5)
myvar2 <- rep(rep(1:50), 50) + rnorm(2500, 0, 5)
newdata <- data.frame(mysample, myvar1, myvar2)

# Prepare a rasclass object using the dataframe and specifying raster properties
object <- new('rasclass')
object <- setRasclassData(newdata, ncols = 50, nrows = 50,
xllcorner = 0, yllcorner = 0, cellsize = 1, NAvalue = -9999,
samplename = 'mysample')

# Summarize the rasclass object
summary(object)
```

---

writeRaster *Load ESRI asciigrid Data for Classification*

Description

This function exports the information in a rasclassRaster object into an external ESRI asciigrid file.

Usage

```
writeRaster(object, path = "predictedGrid.asc")
```

Arguments

object	A rasclassRaster object.
path	Optional path to write ascii file, the default writes a file "predictedGrid.asc" into the workspace folder.

References

ESRI ASCII raster format definition.

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/ESRI_ASCII_raster_format/009t0000000z000000/

See Also

rasclass-package, rasclass-class, rasclassRaster-class,

readRaster,

readRasterFolder, setRasclassData,

buildFormula, checkRasclass,

rasclassMlc, classifyRasclass

Examples

```
## Not run:
writeRaster(myRaster, path = "mypath")

## End(Not run)
```

Index