Package 'RMOA'

February 19, 2015

Version 1.0

Title Connect R with MOA for Massive Online Analysis

Description Connect R with MOA (Massive Online Analysis - http://moa.cms.waikato.ac.nz) to build classification models and regression models on streaming data or out-of-RAM data

Depends RMOAjars (>= 1.0), rJava (>= 0.6-3), methods

Suggests ff

SystemRequirements Java (>= 5.0)

License GPL-3

Copyright Code is Copyright (C) Jan Wijffels and BNOSAC

Maintainer Jan Wijffels <jwijffels@bnosac.be>

URL http://www.bnosac.be, https://github.com/jwijffels/RMOA, http://moa.cms.waikato.ac.nz/

Author Jan Wijffels [aut, cre], BNOSAC [cph]

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-19 00:54:31

R topics documented:

`datastream` *Datastream objects and methods*

Description

Reference object of class datastream. This is a generic class which holds general information about the data stream.
Currently streams are implemented for data in table format (streams of read.table, read.csv, read.csv2, read.delim, read.delim2), data in RAM (data.frame, matrix), data in ff (on disk).
See the documentation of `datastream_file`, `datastream_dataframe`, `datastream_matrix`, and `datastream_ffdf`

Arguments

`description`	The name how the stream is labelled
`args`	a list with arguments used to set up the stream and used in the datastream methods

Value

A class of type datastream which contains

description: character with the name how the stream is labelled.

state: integer with the current state at which the stream will read new instances of data

processed: integer with the number of instances already processed

finished: logical indicating if the stream has finished processing all the instances

args: list with arguments passed on to the stream when it is created (e.g. arguments of read.table)

See Also

`datastream_file`

Examples

```
## Basic example, showing the general methods available for a datastream object
x <- datastream(description = "My own datastream", args = list(a = "TEST"))
x
str(x)
try(x$get_points(x))
```

`datastream_dataframe` *data streams on a data.frame*

Description

Reference object of class `datastream_dataframe`. This is a class which inherits from class `datastream` and which can be used to read in a stream from a data.frame.

Arguments

`data`	a data.frame to extract data from in a streaming way

Value

A class of type `datastream_dataframe` which contains

data: The data.frame to extract instances from

all fields of the datastream superclass: See `datastream`

Methods

- `get_points(n)` Get data from a datastream object.

  n integer, indicating the number of instances to retrieve from the datastream

See Also

`datastream`

Examples

```
x <- datastream_dataframe(data=iris)
x$get_points(10)
x
x$get_points(10)
x
```

`datastream_ffdf` *data streams on an ffdf*

Description

Reference object of class `datastream_ffdf`. This is a class which inherits from class `datastream` and which can be used to read in a stream from a ffdf from the ff package.

Arguments

`data`	a data.frame to extract data from in a streaming way

Value

A class of type `datastream_ffdf` which contains

data: The ffdf to extract instances from

all fields of the datastream superclass: See `datastream`

Methods

- `get_points(n)` Get data from a datastream object.

  n integer, indicating the number of instances to retrieve from the datastream

See Also

`datastream`

Examples

```
## You need to load package ff before you can use datastream_ffdf
require(ff)
irisff <- as.ffdf(factorise(iris))
x <- datastream_ffdf(data=irisff)
x$get_points(10)
x
x$get_points(10)
x
```

`datastream_file` *File data stream*

Description

Reference object of class `datastream_file`. This is a class which inherits from class `datastream` and which can be used to read in a stream from a file. A number of file readers have been implemented, namely `datastream_table`, `datastream_csv`, `datastream_csv2`, `datastream_delim`, `datastream_delim2`.
See the examples.

Arguments

`description`	The name how the stream is labelled
`FUN`	The function to use to read in the file. Defaults to `read.table` for `datastream_table`, `read.csv` for `datastream_csv`, `read.csv2` for `datastream_csv2`, `read.delim` for `datastream_delim`, `read.delim2` for `datastream_delim2`
`columnnames`	optional character vector of column to overwrite the column names of the data read in with in `get_points`
`file`	The file to read in. See e.g. `read.table`
`...`	parameters passed on to `FUN`. See e.g. `read.table`

Value

A class of type `datastream_file` which contains

FUN: The function to use to read in the file

connection: A connection to the file

columnnames: A character vector of column names to overwrite the column names with in `get_points`

all fields of the datastream superclass: See `datastream`

Methods

- `get_points(n)` Get data from a datastream object.

  n integer, indicating the number of instances to retrieve from the datastream

See Also

`read.table`, `read.csv`, `read.csv2`, `read.delim`, `read.delim2`

Examples

```
mydata <- iris
mydata$Species[2:3] <- NA
## Example of a CSV file stream
myfile <- tempfile()
write.csv(iris, file = myfile, row.names=FALSE, na = "")
x <- datastream_csv(file = myfile, na.strings = "")
x
x$get_points(n=10)
x
x$get_points(n=10)
x
x$stop()

## Create your own specific file stream
write.table(iris, file = myfile, row.names=FALSE, na = "")
x <- datastream_file(description="My file defintion stream", FUN=read.table,
 file = myfile, header=TRUE, na.strings="")
x$get_points(n=10)
x
```

---

datastream_matrix *data streams on a matrix*

Description

Reference object of class `datastream_matrix`. This is a class which inherits from class `datastream` and which can be used to read in a stream from a matrix.

Arguments

`data`	a matrix to extract data from in a streaming way

Value

A class of type `datastream_matrix` which contains

data: The matrix to extract instances from

all fields of the datastream superclass: See `datastream`

Methods

- `get_points(n)` Get data from a datastream object.

  n integer, indicating the number of instances to retrieve from the datastream

See Also

`datastream`

Examples

```
data <- matrix(rnorm(1000*10), nrow = 1000, ncol = 10)
x <- datastream_matrix(data=data)
x$get_points(10)
x
x$get_points(10)
x
```

---

factorise *Convert character strings to factors in a dataset*

Description

Convert character strings to factors in a dataset

Usage

```
factorise(x, ...)
```

Arguments

x	object of class data.frame
...	other parameters currently not used yet

Value

a data.frame with the information in x where character columns are converted to factors

Examples

```
data(iris)
str(iris)
mydata <- factorise(iris)
str(mydata)
```

MOAattributes *Define the attributes of a dataset (factor levels, numeric or string data) in a MOA setting*

Description

Define the attributes of a dataset (factor levels, numeric or string data) in a MOA setting

Usage

```
MOAattributes(data, ...)
```

Arguments

data	object of class data.frame
...	other parameters currently not used yet

Value

An object of class MOAmodelAttributes

Examples

```
data(iris)
mydata <- factorise(iris)
atts <- MOAattributes(data=mydata)
atts
```

MOAoptions *Get and set options for models build with MOA.*

Description

Get and set options for models build with MOA.

Usage

```
MOAoptions(model, ...)
```

Arguments

model	character string with a model or an object of class MOA_model. E.g. HoeffdingTree, DecisionStump, NaiveBayes, HoeffdingOptionTree, ... The list of known models can be obtained by typing RMOA:::.moaknownmodels. See the examples.
...	other parameters specifying the MOA modelling options of each model. See the examples.

Value

An object of class MOAmodelOptions.
This is a list with elements:

1. model: The name of the model
2. moamodelname: The purpose of the model known by MOA (getPurposeString)
3. javaObj: a java reference of MOA options
4. options: a list with options of the MOA model. Each list element contains the Name of the option, the Purpose of the option and the current Value

See the examples.

Examples

```
control <- MOAoptions(model = "HoeffdingTree")
control
MOAoptions(model = "HoeffdingTree", leafprediction = "MC",
   removePoorAtts = TRUE, binarySplits = TRUE, tieThreshold = 0.20)

## Other models known by RMOA
RMOA:::.moaknownmodels

## Classification Trees
MOAoptions(model = "AdaHoeffdingOptionTree")
MOAoptions(model = "ASHoeffdingTree")
MOAoptions(model = "DecisionStump")
MOAoptions(model = "HoeffdingAdaptiveTree")
MOAoptions(model = "HoeffdingOptionTree")
MOAoptions(model = "HoeffdingTree")
MOAoptions(model = "LimAttHoeffdingTree")
MOAoptions(model = "RandomHoeffdingTree")
## Classification using Bayes rule
MOAoptions(model = "NaiveBayes")
MOAoptions(model = "NaiveBayesMultinomial")
## Classification using Active learning
MOAoptions(model = "ActiveClassifier")
## Classification using Ensemble learning
MOAoptions(model = "AccuracyUpdatedEnsemble")
MOAoptions(model = "AccuracyWeightedEnsemble")
MOAoptions(model = "ADACC")
MOAoptions(model = "DACC")
MOAoptions(model = "LeveragingBag")
MOAoptions(model = "OCBoost")
MOAoptions(model = "OnlineAccuracyUpdatedEnsemble")
MOAoptions(model = "OzaBag")
MOAoptions(model = "OzaBagAdwin")
MOAoptions(model = "OzaBagASHT")
MOAoptions(model = "OzaBoost")
MOAoptions(model = "OzaBoostAdwin")
MOAoptions(model = "TemporallyAugmentedClassifier")
MOAoptions(model = "WeightedMajorityAlgorithm")

## Regressions
MOAoptions(model = "AMRulesRegressor")
MOAoptions(model = "FadingTargetMean")
MOAoptions(model = "FIMTDD")
MOAoptions(model = "ORTO")
MOAoptions(model = "Perceptron")
MOAoptions(model = "SGD")
MOAoptions(model = "TargetMean")
```

MOA_classification_activelearning

MOA active learning classification

Description

MOA active learning classification

Usage

```
ActiveClassifier(control = NULL, ...)
```

Arguments

control	an object of class MOAmodelOptions as obtained by calling MOAoptions
...	options of parameters passed on to MOAoptions, in case control is left to NULL. Ignored if control is supplied

Value

An object of class MOA_classifier which sets up an untrained MOA model, which can be trained using trainMOA

See Also

MOAoptions, trainMOA

Examples

```
ctrl <- MOAoptions(model = "ActiveClassifier")
mymodel <- ActiveClassifier(control=ctrl)
mymodel
```

MOA_classification_bayes

MOA bayesian classification

Description

MOA bayesian classification

Usage

```
NaiveBayes(control = NULL, ...)

NaiveBayesMultinomial(control = NULL, ...)
```

Arguments

control	an object of class MOAmodelOptions as obtained by calling MOAoptions
...	options of parameters passed on to MOAoptions, in case control is left to NULL. Ignored if control is supplied

Value

An object of class `MOA_classifier` which sets up an untrained MOA model, which can be trained using `trainMOA`

See Also

`MOAoptions`, `trainMOA`

Examples

```
ctrl <- MOAoptions(model = "NaiveBayes")
mymodel <- NaiveBayes(control=ctrl)
mymodel
```

MOA_classification_ensemblelearning

MOA classification using ensembles

Description

MOA classification using ensembles (bagging/boosting/stacking/other)

Usage

```
AccuracyUpdatedEnsemble(control = NULL, ...)

AccuracyWeightedEnsemble(control = NULL, ...)

ADACC(control = NULL, ...)

DACC(control = NULL, ...)

LeveragingBag(control = NULL, ...)

LimAttClassifier(control = NULL, ...)

OCBoost(control = NULL, ...)

OnlineAccuracyUpdatedEnsemble(control = NULL, ...)

OzaBag(control = NULL, ...)

OzaBagAdwin(control = NULL, ...)

OzaBagASHT(control = NULL, ...)

OzaBoost(control = NULL, ...)
```

```
OzaBoostAdwin(control = NULL, ...)

TemporallyAugmentedClassifier(control = NULL, ...)

WeightedMajorityAlgorithm(control = NULL, ...)
```

Arguments

control	an object of class MOAmodelOptions as obtained by calling MOAoptions
...	options of parameters passed on to MOAoptions, in case control is left to NULL. Ignored if control is supplied

Value

An object of class MOA_classifier which sets up an untrained MOA model, which can be trained using trainMOA

See Also

MOAoptions, trainMOA

Examples

```
ctrl <- MOAoptions(model = "OzaBoostAdwin")
mymodel <- OzaBoostAdwin(control=ctrl)
mymodel
```

MOA_classification_trees

MOA classification trees

Description

MOA classification trees

Usage

```
AdaHoeffdingOptionTree(control = NULL, ...)

ASHoeffdingTree(control = NULL, ...)

DecisionStump(control = NULL, ...)

HoeffdingAdaptiveTree(control = NULL, ...)

HoeffdingOptionTree(control = NULL, ...)
```

```
HoeffdingTree(control = NULL, ...)

LimAttHoeffdingTree(control = NULL, ...)

RandomHoeffdingTree(control = NULL, ...)
```

Arguments

control	an object of class MOAmodelOptions as obtained by calling MOAoptions
...	options of parameters passed on to MOAoptions, in case control is left to NULL. Ignored if control is supplied

Value

An object of class MOA_classifier which sets up an untrained MOA model, which can be trained using trainMOA

See Also

MOAoptions, trainMOA

Examples

```
ctrl <- MOAoptions(model = "HoeffdingTree", leafprediction = "MC",
   removePoorAtts = TRUE, binarySplits = TRUE, tieThreshold = 0.20)
hdt <- HoeffdingTree(control=ctrl)
hdt
hdt <- HoeffdingTree(numericEstimator = "GaussianNumericAttributeClassObserver")
hdt
```

MOA_classifier *Create a MOA classifier*

Description

Create a MOA classifier

Usage

```
MOA_classifier(model, control = NULL, ...)
```

Arguments

model	character string with a model. E.g. HoeffdingTree, DecisionStump, NaiveBayes, HoeffdingOptionTree, ... The list of known models can be obtained by typing RMOA:::.moaknownmodels. See the examples and MOAoptions.
control	an object of class MOAmodelOptions as obtained by calling MOAoptions
...	options of parameters passed on to MOAoptions, in case control is left to NULL. Ignored if control is supplied

Value

An object of class `MOA_classifier`

See Also

`MOAoptions`

Examples

```
RMOA:::.moaknownmodels
ctrl <- MOAoptions(model = "HoeffdingTree", leafprediction = "MC",
   removePoorAtts = TRUE, binarySplits = TRUE, tieThreshold = 0.20)
hdt <- MOA_classifier(model = "HoeffdingTree", control=ctrl)
hdt
hdt <- MOA_classifier(
 model = "HoeffdingTree",
 numericEstimator = "GaussianNumericAttributeClassObserver")
hdt
```

`MOA_regressor` *Create a MOA regressor*

Description

Create a MOA regressor

Usage

```
MOA_regressor(model, control = NULL, ...)
```

Arguments

`model`	character string with a model. E.g. AMRulesRegressor, FadingTargetMean, FIMTDD, ORTO, Perceptron, RandomRules, SGD, TargetMean, ... The list of known models can be obtained by typing RMOA:::.moaknownmodels. See the examples and `MOAoptions`.
`control`	an object of class `MOAmodelOptions` as obtained by calling `MOAoptions`
`...`	options of parameters passed on to `MOAoptions`, in case `control` is left to NULL. Ignored if `control` is supplied

Value

An object of class `MOA_regressor`

See Also

`MOAoptions`

Examples

```
mymodel <- MOA_regressor(model = "FIMTDD")
mymodel
data(iris)
iris <- factorise(iris)
irisdatastream <- datastream_dataframe(data=iris)
## Train the model
mytrainedmodel <- trainMOA(model = mymodel,
 Sepal.Length ~ Petal.Length + Species, data = irisdatastream)
mytrainedmodel$model

summary(lm(Sepal.Length ~ Petal.Length + Species, data = iris))
predict(mytrainedmodel, newdata=iris)
```

`MOA_regressors` *MOA regressors*

Description

MOA regressors

Usage

```
FIMTDD(control = NULL, ...)

AMRulesRegressor(control = NULL, ...)
```

Arguments

`control`	an object of class `MOAmodelOptions` as obtained by calling `MOAoptions`
`...`	options of parameters passed on to `MOAoptions`, in case `control` is left to NULL. Ignored if `control` is supplied

Value

An object of class `MOA_classifier` which sets up an untrained MOA model, which can be trained using `trainMOA`

See Also

`MOAoptions`, `trainMOA`

Examples

```
ctrl <- MOAoptions(model = "FIMTDD", DoNotDetectChanges = TRUE, noAnomalyDetection=FALSE,
   univariateAnomalyprobabilityThreshold = 0.5, verbosity = 5)
mymodel <- FIMTDD(control=ctrl)
mymodel
mymodel <- FIMTDD(ctrlDoNotDetectChanges = FALSE)
mymodel
```

predict.MOA_trainedmodel

Predict using a MOA classifier on a new dataset

Description

Predict using a MOA classifier on a new dataset. Make sure the new dataset has the same structure and the same levels as `get_points` returns on the datastream which was used in `trainMOA`

Usage

```
## S3 method for class 'MOA_trainedmodel'
predict(object, newdata, type = "response",
  transFUN = object$transFUN, ...)
```

Arguments

`object`	an object of class `MOA_trainedmodel`, as returned by `trainMOA`
`newdata`	a data.frame with the same structure and the same levels as used in `trainMOA`
`type`	a character string, either 'response' or 'votes'
`transFUN`	a function which is used on `newdata` before applying `model.frame`. Useful if you want to change the results `get_points` on the datastream (e.g. for making sure the factor levels are the same in each chunk of processing, some data cleaning, ...). Defaults to `transFUN` available in `object`.
`...`	other arguments, currently not used yet

Value

A matrix of votes or a vector with the predicted class

See Also

`trainMOA`

Examples

```
## Hoeffdingtree
hdt <- HoeffdingTree(numericEstimator = "GaussianNumericAttributeClassObserver")
data(iris)
## Make a training set
iris <- factorise(iris)
traintest <- list()
traintest$trainidx <- sample(nrow(iris), size=nrow(iris)/2)
traintest$trainingset <- iris[traintest$trainidx, ]
traintest$testset <- iris[-traintest$trainidx, ]
irisdatastream <- datastream_dataframe(data=traintest$trainingset)
## Train the model
```

```
hdtreetrained <- trainMOA(model = hdt,
 Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
 data = irisdatastream)

## Score the model on the holdoutset
scores <- predict(hdtreetrained,
  newdata=traintest$testset[, c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width")],
   type="response")
str(scores)
table(scores, traintest$testset$Species)
scores <- predict(hdtreetrained, newdata=traintest$testset, type="votes")
head(scores)
```

`summary.MOA_classifier` *Summary statistics of a MOA classifier*

Description

Summary statistics of a MOA classifier

Usage

```
## S3 method for class 'MOA_classifier'
summary(object, ...)
```

Arguments

`object`	an object of class `MOA_classifier`
`...`	other arguments, currently not used yet

Value

the form of the return value depends on the type of MOA model

Examples

```
hdt <- HoeffdingTree(numericEstimator = "GaussianNumericAttributeClassObserver")
hdt
data(iris)
iris <- factorise(iris)
irisdatastream <- datastream_dataframe(data=iris)
## Train the model
hdtreetrained <- trainMOA(model = hdt,
 Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
 data = irisdatastream)
summary(hdtreetrained$model)
```

summary.MOA_regressor *Summary statistics of a MOA regressor*

Description

Summary statistics of a MOA regressor

Usage

```
## S3 method for class 'MOA_regressor'
summary(object, ...)
```

Arguments

object	an object of class MOA_regressor
...	other arguments, currently not used yet

Value

the form of the return value depends on the type of MOA model

Examples

```
## TODO
```

trainMOA *Train a MOA classifier (e.g. a HoeffdingTree) on a datastream*

Description

Train a MOA classifier (e.g. a HoeffdingTree) on a datastream

Usage

```
trainMOA(model, formula, data, subset, na.action = na.exclude,
  transFUN = identity, chunksize = 1000, reset = TRUE, trace = FALSE,
  options = list(maxruntime = +Inf))
```

Arguments

model	an object of class MOA_model, as returned by MOA_classifier, e.g. a HoeffdingTree
formula	a symbolic description of the model to be fit.
data	an object of class datastream set up e.g. with datastream_file, datastream_dataframe, datastream_matrix, datastream_ffdf or your own datastream.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. See model.frame for details. Defaults to na.exclude.
transFUN	a function which is used after obtaining chunksize number of rows from the data datastream before applying model.frame. Useful if you want to change the results get_points on the datastream (e.g. for making sure the factor levels are the same in each chunk of processing, some data cleaning, ...). Defaults to identity.
chunksize	the number of rows to obtain from the data datastream in one chunk of model processing. Defaults to 1000. Can be used to speed up things according to the backbone architecture of the datastream.
reset	logical indicating to reset the MOA_classifier so that it forgets what it already has learned. Defaults to TRUE.
trace	logical, indicating to show information on how many datastream chunks are already processed as a message.
options	a names list of further options. Currently not used.

Value

An object of class MOA_trainedmodel which is a list with elements

- model: the updated supplied model object of class MOA_classifier
- call: the matched call
- na.action: the vatlue of na.action
- terms: the terms in the model
- transFUN: the transFUN argument

See Also

MOA_classifier, datastream_file, datastream_dataframe, datastream_matrix, datastream_ffdf, datastream, predict.MOA_trainedmodel

Examples

```
hdt <- HoeffdingTree(numericEstimator = "GaussianNumericAttributeClassObserver")
hdt
data(iris)
iris <- factorise(iris)
irisdatastream <- datastream_dataframe(data=iris)
```

```
irisdatastream$get_points(3)

mymodel <- trainMOA(model = hdt, Species ~ Sepal.Length + Sepal.Width + Petal.Length,
 data = irisdatastream, chunksize = 10)
mymodel$model
irisdatastream$reset()
mymodel <- trainMOA(model = hdt,
 Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Length^2,
 data = irisdatastream, chunksize = 10, reset=TRUE, trace=TRUE)
mymodel$model
```

Index