

# Package ‘RMOA’

July 17, 2022

**Version** 1.1.0

**Title** Connect R with MOA for Massive Online Analysis

**Description** Connect R with MOA (Massive Online Analysis - [<https://moa.cms.waikato.ac.nz/>](https://moa.cms.waikato.ac.nz/)) to build classification models and regression models on streaming data or out-of-RAM data. Also streaming recommendation models are made available.

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

**Suggests** ff, recommenderlab

**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>,  
<https://moa.cms.waikato.ac.nz/>

**RoxygenNote** 7.1.1

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2022-07-17 21:00:02 UTC

## R topics documented:

datastream . . . . .	2
datastream_dataframe . . . . .	3
datastream_ffdf . . . . .	4
datastream_file . . . . .	5
datastream_matrix . . . . .	6
factorise . . . . .	7
MOAattributes . . . . .	8

MOAoptions	8
MOA_classification_activelearning	10
MOA_classification_bayes	11
MOA_classification_ensemblelearning	11
MOA_classification_trees	13
MOA_classifier	14
MOA_recommendation_engines	15
MOA_recommender	15
MOA_regressor	16
MOA_regressors	17
predict.MOA_trainedmodel	18
summary.MOA_classifier	20
summary.MOA_recommender	21
summary.MOA_regressor	22
trainMOA	22
trainMOA.MOA_classifier	23
trainMOA.MOA_recommender	24
trainMOA.MOA_regressor	26

## Index 28

---

datastream	<i>Datastream objects and methods</i>
------------	---------------------------------------

---

### 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 ffd*

---

### 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	<i>File data stream</i>
-----------------	-------------------------

---

### 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

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

### 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 definition stream", FUN=read.table,
  file = myfile, header=TRUE, na.strings="")
x$get_points(n=10)
x
x$stop()

## Clean up for CRAN
file.remove(myfile)

```

---

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	<i>Define the attributes of a dataset (factor levels, numeric or string data) in a MOA setting</i>
---------------	--

---

**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	<i>Get and set options for models build with MOA.</i>
------------	---

---

**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")

## Recommendation engines
MOAoptions(model = "BRISMFPredictor")
MOAoptions(model = "BaselinePredictor")
```

---

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 <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , 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 <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , 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 <code>MOAmodelOptions</code> as obtained by calling <code>MOAoptions</code>
...	options of parameters passed on to <code>MOAoptions</code> , in case control is left to <code>NULL</code> . 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, ...)
ASHHoeffdingTree(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 <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , 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, Naive-Bayes, HoeffdingOptionTree, ... The list of known models can be obtained by typing <code>RMOA:::moaknownmodels</code> . See the examples and <a href="#">MOAoptions</a> .
control	an object of class MOAmodelOptions as obtained by calling <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , 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\_recommendation\_engines  
*MOA recommendation engines*

---

**Description**

MOA recommendation engines

**Usage**

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

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

**Arguments**

control	an object of class MOAmodelOptions as obtained by calling <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , in case control is left to NULL. Ignored if control is supplied

**Value**

An object of class MOA\_recommender which sets up an untrained MOA model, which can be trained using [trainMOA](#)

**See Also**

[MOAoptions](#), [trainMOA](#)

**Examples**

```
ctrl <- MOAoptions(model = "BRISMFPPredictor", features = 10)
brism <- BRISMFPPredictor(control=ctrl)
brism
baseline <- BaselinePredictor()
baseline
```

---

MOA\_recommender      *Create a MOA recommendation engine*

---

**Description**

Create a MOA recommendation engine

**Usage**

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

**Arguments**

model	character string with a model. E.g. BRISMFPredictor, BaselinePredictor The list of known models can be obtained by typing RMOA:::moaknownmodels. See the examples and <a href="#">MOAoptions</a> .
control	an object of class MOAmodelOptions as obtained by calling <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , in case control is left to NULL. Ignored if control is supplied

**Value**

An object of class MOA\_recommender

**See Also**

[MOAoptions](#)

**Examples**

```
RMOA:::moaknownmodels
ctrl <- MOAoptions(model = "BRISMFPredictor", features = 10, lRate=0.002)
brism <- MOA_recommender(model = "BRISMFPredictor", control=ctrl)
brism
MOAoptions(model = "BaselinePredictor")
baseline <- MOA_recommender(model = "BaselinePredictor")
baseline
```

---

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 <a href="#">MOAoptions</a> .
control	an object of class MOAmodelOptions as obtained by calling <a href="#">MOAoptions</a>
...	options of parameters passed on to <a href="#">MOAoptions</a> , 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**

```

TargetMean(control = NULL, ...)

FadingTargetMean(control = NULL, ...)

Perceptron(control = NULL, ...)

AMRulesRegressor(control = NULL, ...)

FIMTDD(control = NULL, ...)

ORTO(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, MOA regressor or MOA recommender on a new dataset*

---

**Description**

Predict using a MOA classifier, MOA regressor or MOA recommender 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, na.action = na.fail, ...)
```

**Arguments**

<code>object</code>	an object of class <code>MOA_trainedmodel</code> , as returned by <code>trainMOA</code>
<code>newdata</code>	a <code>data.frame</code> with the same structure and the same levels as used in <code>trainMOA</code> for MOA classifier, MOA regressor, a <code>data.frame</code> with at least the user/item columns which were used in <code>trainMOA</code> when training the MOA recommendation engine
<code>type</code>	a character string, either 'response' or 'votes'
<code>transFUN</code>	a function which is used on <code>newdata</code> before applying <code>model.frame</code> . Useful if you want to change the results <code>get_points</code> on the datastream (e.g. for making sure the factor levels are the same in each chunk of processing, some data cleaning, ...). Defaults to <code>transFUN</code> available in <code>object</code> .
<code>na.action</code>	passed on to <code>model.frame</code> when constructing the <code>model.matrix</code> from <code>newdata</code> . Defaults to <code>na.fail</code> .
<code>...</code>	other arguments, currently not used yet

**Value**

A matrix of votes or a vector with the predicted class for MOA classifier or MOA regressor. A

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

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

## Prediction based on recommendation engine
require(recommenderlab)
data(MovieLense)
x <- getData.frame(MovieLense)
x$itemid <- as.integer(as.factor(x$item))
x$userid <- as.integer(as.factor(x$user))
x$rating <- as.numeric(x$rating)
x <- head(x, 2000)

movielensestream <- datastream_dataframe(data=x)
movielensestream$get_points(3)

ctrl <- MOAoptions(model = "BRISMFpredictor", features = 10)
brism <- BRISMFpredictor(control=ctrl)
mymodel <- trainMOA(model = brism, rating ~ userid + itemid,
  data = movielensestream, chunksize = 1000, trace=TRUE)

overview <- summary(mymodel$model)
str(overview)
```

```
predict(mymodel, head(x, 10), type = "response")

x <- expand.grid(userid=overview$users[1:10], itemid=overview$items)
predict(mymodel, x, type = "response")
```

---

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

---

`summary.MOA_recommender`*Summary statistics of a MOA recommender*

---

## Description

Summary statistics of a MOA recommender

## Usage

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

## Arguments

<code>object</code>	an object of class <code>MOA_recommender</code>
<code>...</code>	other arguments, currently not used yet

## Value

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

## Examples

```
require(recommenderlab)  
data(MovieLens)  
x <- getData.frame(MovieLens)  
x$itemid <- as.integer(as.factor(x$item))  
x$userid <- as.integer(as.factor(x$user))  
x$rating <- as.numeric(x$rating)  
x <- head(x, 2000)  
  
movielensestream <- datastream_dataframe(data=x)  
movielensestream$get_points(3)  
  
ctrl <- MOAoptions(model = "BRISMFpredictor", features = 10)  
brism <- BRISMFpredictor(control=ctrl)  
mymodel <- trainMOA(model = brism, rating ~ userid + itemid,  
  data = movielensestream, chunksize = 1000, trace=TRUE)  
  
overview <- summary(mymodel$model)  
str(overview)  
predict(mymodel, head(x, 10), type = "response")
```

---

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/regressor/recommendation engine on a datastream*

---

### Description

Train a MOA classifier/regressor/recommendation engine on a datastream

### Usage

```
trainMOA(model, ...)
```

### Arguments

model            an object of class MOA\_model, as returned by [MOA\\_classifier](#), [MOA\\_regressor](#), [MOA\\_recommender](#)  
 ...              other parameters passed on to the methods

### Value

An object of class MOA\_trainedmodel which is returned by the methods for the specific model. See [trainMOA.MOA\\_classifier](#), [trainMOA.MOA\\_regressor](#), [trainMOA.MOA\\_recommender](#)

**See Also**

[trainMOA.MOA\\_classifier](#), [trainMOA.MOA\\_regressor](#), [trainMOA.MOA\\_recommender](#)

---

trainMOA.MOA\_classifier

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

---

**Description**

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

**Usage**

```
## S3 method for class 'MOA_classifier'
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 <code>MOA_model</code> , as returned by <a href="#">MOA_classifier</a> , e.g. a <a href="#">HoeffdingTree</a>
formula	a symbolic description of the model to be fit.
data	an object of class <code>datastream</code> set up e.g. with <a href="#">datastream_file</a> , <a href="#">datastream_dataframe</a> , <a href="#">datastream_matrix</a> , <a href="#">datastream_ffdf</a> 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 <a href="#">model.frame</a> for details. Defaults to <a href="#">na.exclude</a> .
transFUN	a function which is used after obtaining <code>chunksize</code> number of rows from the data datastream before applying <a href="#">model.frame</a> . Useful if you want to change the results <code>get_points</code> on the datastream (e.g. for making sure the factor levels are the same in each chunk of processing, some data cleaning, ...). Defaults to <a href="#">identity</a> .
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 <code>MOA_classifier</code> 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.
...	other arguments, currently not used yet

**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 value 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
```

---

```
trainMOA.MOA_recommender
```

*Train a MOA recommender (e.g. a BRISMFPredictor) on a datastream*

---

**Description**

Train a MOA recommender (e.g. a BRISMFPredictor) on a datastream

**Usage**

```
## S3 method for class 'MOA_recommender'
trainMOA(model, formula, data, subset,
  na.action = na.exclude, transFUN = identity, chunksize = 1000,
  trace = FALSE, options = list(maxruntime = +Inf), ...)
```



**Arguments**

model	an object of class <code>MOA_model</code> , as returned by <code>MOA_recommender</code> , e.g. a <code>BRISMFPredictor</code>
formula	a symbolic description of the model to be fit. This should be of the form <code>rating ~ userid + itemid</code> , in that sequence. These should be columns in the data, where <code>userid</code> and <code>itemid</code> are integers and <code>rating</code> is numeric.
data	an object of class <code>datastream</code> set up e.g. with <code>datastream_file</code> , <code>datastream_dataframe</code> , <code>datastream_matrix</code> , <code>datastream_ffdf</code> or your own <code>datastream</code> .
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 <code>model.frame</code> for details. Defaults to <code>na.exclude</code> .
transFUN	a function which is used after obtaining <code>chunksize</code> number of rows from the data <code>datastream</code> before applying <code>model.frame</code> . Useful if you want to change the results <code>get_points</code> on the <code>datastream</code> (e.g. for making sure the factor levels are the same in each chunk of processing, some data cleaning, ...). Defaults to <code>identity</code> .
chunksize	the number of rows to obtain from the data <code>datastream</code> in one chunk of model processing. Defaults to 1000. Can be used to speed up things according to the backbone architecture of the <code>datastream</code> .
trace	logical, indicating to show information on how many <code>datastream</code> chunks are already processed as a message.
options	a names list of further options. Currently not used.
...	other arguments, currently not used yet

**Value**

An object of class `MOA_trainedmodel` which is a list with elements

- `model`: the updated supplied `model` object of class `MOA_recommender`
- `call`: the matched call
- `na.action`: the value of `na.action`
- `terms`: the terms in the model
- `transFUN`: the `transFUN` argument

**See Also**

[MOA\\_recommender](#), [datastream\\_file](#), [datastream\\_dataframe](#), [datastream\\_matrix](#), [datastream\\_ffdf](#), [datastream](#), [predict.MOA\\_trainedmodel](#)

**Examples**

```
require(recommenderlab)
data(MovieLens)
x <- getData.frame(MovieLens)
x$itemid <- as.integer(as.factor(x$itemid))
```

```
x$userid <- as.integer(as.factor(x$user))
x$rating <- as.numeric(x$rating)
x <- head(x, 5000)

movielensestream <- datastream_dataframe(data=x)
movielensestream$get_points(3)

ctrl <- MOAoptions(model = "BRISMFPredictor", features = 10)
brism <- BRISMFPredictor(control=ctrl)
mymodel <- trainMOA(model = brism, rating ~ userid + itemid,
  data = movielensestream, chunksize = 1000, trace=TRUE)
summary(mymodel$model)
```

---

```
trainMOA.MOA_regressor
```

*Train a MOA regressor (e.g. a FIMTDD) on a datastream*

---

## Description

Train a MOA regressor (e.g. a FIMTDD) on a datastream

## Usage

```
## S3 method for class 'MOA_regressor'
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 <code>MOA_model</code> , as returned by <code>MOA_regressor</code> , e.g. a <code>FIMTDD</code>
formula	a symbolic description of the model to be fit.
data	an object of class <code>datastream</code> set up e.g. with <code>datastream_file</code> , <code>datastream_dataframe</code> , <code>datastream_matrix</code> , <code>datastream_ffdf</code> 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 <code>model.frame</code> for details. Defaults to <code>na.exclude</code> .
transFUN	a function which is used after obtaining <code>chunksize</code> number of rows from the data datastream before applying <code>model.frame</code> . Useful if you want to change the results <code>get_points</code> on the datastream (e.g. for making sure the factor levels are the same in each chunk of processing, some data cleaning, ...). Defaults to <code>identity</code> .
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_regressor 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.
...	other arguments, currently not used yet

**Value**

An object of class MOA\_trainedmodel which is a list with elements

- model: the updated supplied model object of class MOA\_regressor
- call: the matched call
- na.action: the value of na.action
- terms: the terms in the model
- transFUN: the transFUN argument

**See Also**

[MOA\\_regressor](#), [datastream\\_file](#), [datastream\\_dataframe](#), [datastream\\_matrix](#), [datastream\\_ffdf](#), [datastream](#), [predict.MOA\\_trainedmodel](#)

**Examples**

```

mymodel <- MOA_regressor(model = "FIMTDD")
mymodel
data(iris)
iris <- factorise(iris)
irisdatastream <- datastream_dataframe(data=iris)
irisdatastream$get_points(3)
## Train the model
mytrainedmodel <- trainMOA(model = mymodel,
  Sepal.Length ~ Petal.Length + Species, data = irisdatastream)
mytrainedmodel$model
irisdatastream$reset()
mytrainedmodel <- trainMOA(model = mytrainedmodel$model,
  Sepal.Length ~ Petal.Length + Species, data = irisdatastream,
  chunksize = 10, reset=FALSE, trace=TRUE)
mytrainedmodel$model

```

# Index

- AccuracyUpdatedEnsemble  
(MOA\_classification\_ensemblelearning),  
11
- AccuracyWeightedEnsemble  
(MOA\_classification\_ensemblelearning),  
11
- ActiveClassifier  
(MOA\_classification\_activelearning),  
10
- ADACC  
(MOA\_classification\_ensemblelearning),  
11
- AdaHoeffdingOptionTree  
(MOA\_classification\_trees), 13
- AMRulesRegressor (MOA\_regressors), 17
- ASHoeffdingTree  
(MOA\_classification\_trees), 13
- BaselinePredictor  
(MOA\_recommendation\_engines),  
15
- BRISMFPredictor, 25
- BRISMFPredictor  
(MOA\_recommendation\_engines),  
15
- DACC  
(MOA\_classification\_ensemblelearning),  
11
- datastream, 2, 3–7, 23–27
- datastream\_csv (datastream\_file), 5
- datastream\_csv2 (datastream\_file), 5
- datastream\_dataframe, 2, 3, 23–27
- datastream\_delim (datastream\_file), 5
- datastream\_delim2 (datastream\_file), 5
- datastream\_ffdf, 2, 4, 23–27
- datastream\_file, 2, 3, 5, 23–27
- datastream\_matrix, 2, 6, 23–27
- datastream\_table (datastream\_file), 5
- DecisionStump  
(MOA\_classification\_trees), 13
- factorise, 7
- FadingTargetMean (MOA\_regressors), 17
- FIMTDD, 26
- FIMTDD (MOA\_regressors), 17
- HoeffdingAdaptiveTree  
(MOA\_classification\_trees), 13
- HoeffdingOptionTree  
(MOA\_classification\_trees), 13
- HoeffdingTree, 23
- HoeffdingTree  
(MOA\_classification\_trees), 13
- identity, 23, 25, 26
- LeveragingBag  
(MOA\_classification\_ensemblelearning),  
11
- LimAttClassifier  
(MOA\_classification\_ensemblelearning),  
11
- LimAttHoeffdingTree  
(MOA\_classification\_trees), 13
- MOA\_classification\_activelearning, 10
- MOA\_classification\_bayes, 11
- MOA\_classification\_ensemblelearning,  
11
- MOA\_classification\_trees, 13
- MOA\_classifier, 14, 22–24
- MOA\_recommendation\_engines, 15
- MOA\_recommender, 15, 22, 25
- MOA\_regressor, 16, 22, 26, 27
- MOA\_regressors, 17
- MOAattributes, 8
- MOAoptions, 8, 10–18
- model.frame, 18, 23, 25, 26

na.exclude, [23](#), [25](#), [26](#)  
NaiveBayes (MOA\_classification\_bayes),  
    [11](#)  
NaiveBayesMultinomial  
    (MOA\_classification\_bayes), [11](#)  
OCBoost  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
OnlineAccuracyUpdatedEnsemble  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
ORTO (MOA\_regressors), [17](#)  
OzaBag  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
OzaBagAdwin  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
OzaBagASHT  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
OzaBoost  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
OzaBoostAdwin  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
Perceptron (MOA\_regressors), [17](#)  
predict.MOA\_trainedmodel, [18](#), [24](#), [25](#), [27](#)  
RandomHoeffdingTree  
    (MOA\_classification\_trees), [13](#)  
read.csv, [5](#)  
read.csv2, [5](#)  
read.delim, [5](#)  
read.delim2, [5](#)  
read.table, [5](#)  
summary.MOA\_classifier, [20](#)  
summary.MOA\_recommender, [21](#)  
summary.MOA\_regressor, [22](#)  
TargetMean (MOA\_regressors), [17](#)  
TemporallyAugmentedClassifier  
    (MOA\_classification\_ensemblelearning),  
    [11](#)  
trainMOA, [10–13](#), [15](#), [18](#), [19](#), [22](#)  
trainMOA.MOA\_classifier, [22](#), [23](#), [23](#)  
trainMOA.MOA\_recommender, [22](#), [23](#), [24](#)  
trainMOA.MOA\_regressor, [22](#), [23](#), [26](#)  
WeightedMajorityAlgorithm  
    (MOA\_classification\_ensemblelearning),  
    [11](#)