Package 'rfml'

August 29, 2016

Type Package

Title MarkLogic NoSQL Database Server in-Database Analytics for R

Version 0.1.0

Author Mats Stellwall [aut, cre],
Abdulla Abdurakhmanov [ctb] (Code in xml2json.sjs is from https://code.google.com/p/x2js/)

Maintainer Mats Stellwall <mats.stellwall@gmail.com>

Description Functionality required to efficiently use R with MarkLogic NoSQL Database Server, <http://www.marklogic.com/what-is-marklogic/>. Many basic and complex R operations are pushed down into the database, which removes the main memory boundary of R and allows to make full use of MarkLogic server. In order to use the package you need a MarkLogic Server version 8 or higher.

URL https://github.com/mstellwa/rfml

BugReports https://github.com/mstellwa/rfml/issues

License GPL-3

LazyData TRUE

Imports stats, utils, methods, httr, PKI, jsonlite, XML

Suggests arules, testthat, knitr, rmarkdown

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2016-03-19 14:33:13

R topics documented:

`arith` *Arithmetic Operators*

Description

Arithmetic Operators

Usage

```
## S4 method for signature 'ml.col.def,ml.col.def'
Arith(e1, e2)

## S4 method for signature 'ml.col.def,ANY'
Arith(e1, e2)

## S4 method for signature 'ANY,ml.col.def'
Arith(e1, e2)
```

Arguments

`e1, e2`	numeric vectors or string or ml.col.def-class object.

`as.character,ml.col.def-method`
Cast a ml.col.def-class expresion to string

Description

This function will add a function to cast the expresion of the ml.col.def-class to a string value. The cast will be done when the result is returned to the client.

Usage

```
## S4 method for signature 'ml.col.def'
as.character(x)
```

Arguments

`x`	an ml.col.def-class object

`as.data.frame,ml.data.frame-method`

Pull data from MarkLogic server based on a ml.data.frame object and return it as a data.frame.

Description

Pull data from MarkLogic server based on a ml.data.frame object and return it as a data.frame.

Usage

```
## S4 method for signature 'ml.data.frame'
as.data.frame(x, max.rows = NULL, ...)
```

Arguments

`x`	a ml.data.frame object
`max.rows`	maximum rows to return. Default all rows.
`...`	Not used.

See Also

`ml.data.frame`, `as.ml.data.frame` for uploading data, `rm.ml.data.frame` for delete uploaded data

Examples

```
## Not run:
 library(rfml)
 localConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(localConn, "setosa")
 lIris <- as.data.frame(mlIris)

## End(Not run)
```

`as.integer,ml.col.def-method`

Cast a ml.col.def-class expresion to integer

Description

This function will add a function to cast the expresion of the ml.col.def-class to a integer value. The cast will be done when the result is returned to the client.

Usage

```
## S4 method for signature 'ml.col.def'
as.integer(x)
```

Arguments

x	an ml.col.def-class object

`as.ml.data.frame` *Upload data in a data.frame object or create data based on a ml.data.frame object*

Description

The function will upload the data within a data.frame object or create data in MarkLogic Server based on a ml.data.frame object. Data created based on ml.data.frame will be flat and fields will have the same names as in the .col.name slot. See details for more information about how data is created.

Usage

```
as.ml.data.frame(conn, x, name, format = "json", directory = "")
```

Arguments

conn	A ml.conn object that has a valid connection to a MarkLogic Server
x	a Data Frame or ml.data.frame object.
name	The name of the object.
format	The format od the documents that is created, json or XML. Default is json
directory	The directory to save the documents, needs to start and end with a /. Default saved to /rfml/[username]/[name]/

Details

When data is uploaded or created it will be stored as json documents default, the format parameter controls, and Document URIs, the identifier of a document, is generated based on the string "rfml", the rowname if a data.frame or a counter if it is a ml.data.frame, the loged in username and the name parameter, for example /rfml/admin/iris/. The documents will also belong to a collection named after tne name parameter.

Value

A ml.data.frame object.

See Also

`ml.data.frame`, `as.data.frame` for pulling data, `rm.ml.data.frame` for delete uploaded data

Examples

```
## Not run:
 library(rfml)
 ml.connect()
 # create a ml.data.frame based on the iris data set
 mlIris <- as.ml.data.frame(iris, "iris")

## End(Not run)
```

`as.numeric,ml.col.def-method`

Cast a ml.col.def-class expresion to numeric.

Description

This function will add a function to cast the expresion of the ml.col.def-class to a numeric value. The cast will be done when the result is returned to the client.

Usage

```
## S4 method for signature 'ml.col.def'
as.numeric(x)
```

Arguments

x	an ml.col.def-class object.

`colnames,ml.data.frame-method`

Column Names of an ml.data.frame object

Description

Column Names of an ml.data.frame object

Usage

```
## S4 method for signature 'ml.data.frame'
colnames(x)
```

Arguments

x	an ml.data.frame object

`Compare,ml.col.def,ANY-method`
Relational Operators

Description

Relational operators used for field level filtering of a ml.data.frame object.

Usage

```
## S4 method for signature 'ml.col.def,ANY'
Compare(e1, e2)
```

Arguments

`e1`	an ml.col.def-class object.
`e2`	any object

`cor,ml.col.def,ml.col.def-method`
Correlation

Description

Returns the Pearson correlation coefficient between two ml.data.frame fields.

Usage

```
## S4 method for signature 'ml.col.def,ml.col.def'
cor(x, y = NULL, use = NULL,
  method = NULL)
```

Arguments

`x`	a ml.data.frame field.
`y`	a ml.data.frame field
`use`	not used currently
`method`	not used currently

Details

The function eliminates all pairs for which either the first element or the second element is empty. After the elimination, if the length of the input is less than 2, the function returns the empty sequence. After the elimination, if the standard deviation of the first column or the standard deviation of the second column is 0, the function returns the empty sequence.

Value

The correlation coefficient

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # return the correlation
 cor(mlIris$Sepal.Length, mlIris$Petal.Length)

## End(Not run)
```

---

cor,ml.data.frame,ANY-method

Correlation Matrix

Description

Returns the Pearson correlation coefficient matrix of all numeric fields in a ml.data.frame

Usage

```
## S4 method for signature 'ml.data.frame,ANY'
cor(x, y = NULL, use = NULL, method = NULL)
```

Arguments

x	a ml.data.frame
y	not used when doing a matrix
use	not implemented
method	not implemented

Details

The function eliminates all fields pairs for which either the first element or the second element is empty. After the elimination, if the length of the input is less than 2, the function returns the empty sequence. After the elimination, if the standard deviation of the first column or the standard deviation of the second column is 0, the function returns the empty sequence.

Value

The correlation coefficient matrix

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # return the correlation matrix
 cor(mlIris)

## End(Not run)
```

`cot` *Cotangent*

Description

Returns the cotangent of x.

Usage

```
cot(x)
```

Arguments

x	an ml.col.def-class object.

Value

The cotangent of x.

`cov,ml.col.def,ml.col.def-method` *Covariance*

Description

Returns the sample covariance of two variables, ml.data.frame fields.

Usage

```
## S4 method for signature 'ml.col.def,ml.col.def'
cov(x, y = NULL, use = NULL,
  method = NULL)
```

Arguments

x	a ml.data.frame field.
y	a ml.data.frame field
use	not implemented
method	not implemented

Details

The function eliminates all pairs for which either the first element or the second element is empty. After the elimination, if the length of the input is less than 2, the function returns the empty sequence.

Value

The sample covariance

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn,collection = "iris")
 # return the Covariance
 cov(mlIris$Sepal.Length, mlIris$Petal.Length)

## End(Not run)
```

---

cov.pop *Population Covariance*

Description

Returns the population covariance of two variables, ml.data.frame fields.

Usage

```
cov.pop(x, y)
```

Arguments

x	a ml.data.frame field.
y	a ml.data.frame field

Details

The function eliminates all pairs for which either the first element or the second element is empty. After the elimination, if the length of the input is 0, the function returns the empty sequence.

Value

The population covariance

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # return the population covariance
 cov.pop(mlIris$Sepal.Length, mlIris$Petal.Length)

## End(Not run)
```

---

degrees *Degrees*

Description

Returns numeric expression converted from radians to degrees. The function is applied when the result is returned to the client.

Usage

```
degrees(x)
```

Arguments

x	an ml.col.def-class object.

Value

numeric expression converted from radians to degrees.

---

dim,ml.data.frame-method *Dimensions of an ml.data.frame object*

Description

Dimensions of an ml.data.frame object

Usage

```
## S4 method for signature 'ml.data.frame'
dim(x)
```

Arguments

x	an ml.data.frame object

`head,ml.data.frame-method` *Return the First Part of an ml.data.frame*

Description

Return the First Part of an ml.data.frame

Usage

```
## S4 method for signature 'ml.data.frame'
head(x, n = 6, ...)
```

Arguments

x	an ml.data.frame object
n	a single integer. The number of rows to return, default is 6
...	not used

`is.ml.col.def` *Check if an object is of type ml.col.def-class*

Description

This function checks if the input is of type ml.col.def-class.

Usage

```
is.ml.col.def(x)
```

Arguments

x	The input can be of any type.

Value

True if it is a ml.col.def-class. False otherwise.

is.ml.data.frame *Check if an object is of type ml.data.frame*

Description

This function checks if the input is of type ml.data.frame.

Usage

```
is.ml.data.frame(x)
```

Arguments

x	The input can be of any type.

Value

True if it is a ml.data.frame object. False otherwise.

Math,ml.col.def-method *Miscellaneous Mathematical Functions*

Description

Mathematical functions that can be used on ml.data.frame fields. The function is applied when the result is returned to the client. Only abs, acos, asin, atan, ceiling, cos, cosh, exp, floor, log, log10 , tan, tanh, sqrt, sin, sinh and trunc is currently supported.

Usage

```
## S4 method for signature 'ml.col.def'
Math(x)
```

Arguments

x	an ml.col.def-class object.

`max,ml.col.def-method` *Max*

Description

Returns the maximum value of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
max(x, na.rm = FALSE)
```

Arguments

`x`	a ml.data.frame field.
`na.rm`	not currently used.

Value

The maximum value

Examples

```
## Not run:
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # max
 max(mlIris$Sepal.Length)

## End(Not run)
```

`mean,ml.col.def-method`
Mean

Description

Returns the mean of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
mean(x, na.rm = FALSE)
```

Arguments

x	a ml.data.frame field.
na.rm	not currently used.

Value

The mean

Examples

```
## Not run:
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # mean
 mean(mlIris$Sepal.Length)

## End(Not run)
```

---

median,ml.col.def-method
Median

Description

Returns the median of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
median(x, na.rm = FALSE)
```

Arguments

x	a ml.data.frame field.
na.rm	not currently used.

Value

The median

Examples

```
## Not run:
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # median
 median(mlIris$Sepal.Length)

## End(Not run)
```

`min,ml.col.def-method` *Min*

Description

Returns the minimum value of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
min(x, na.rm = FALSE)
```

Arguments

`x`	a ml.data.frame field.
`na.rm`	not currently used.

Value

The minimum value

Examples

```
## Not run:
 ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(collection = "iris")
 # min
 min(mlIris$Sepal.Length)

## End(Not run)
```

`ml.add.index` *Creates or updates a Range element index.*

Description

The function creates or updates a range element index on the underlying element/property of a ml.data.frame field. The user that is used for the login needs the manage-admin role, or the following privilege:

- http://marklogic.com/xdmp/privileges/manage-admin

Usage

```
ml.add.index(x, scalarType = "string",
  collation = "http://marklogic.com/collation/", namespaceUri = "",
  database = "Documents", host = "", port = "8002", adminuser = "",
  password = "", conn = NA)
```

Arguments

`x`	a ml.data.frame field that the index will be created on
`scalarType`	An atomic type specification. "string" is default
`collation`	For scalarType = string, you can use a different collation than the default. Default is "http://marklogic.com/collation/"
`namespaceUri`	The namespace URI of the XML element, if JSON ignore. Default is empty.
`database`	The name of the database to create the index in. "Documents" is default.
`host`	The hostname or ipadress of the MarkLogic Manage server. Default is the same as used for conn
`port`	The port number of the MarkLogic Manage server. 8002 is used default
`adminuser`	The username of a user that have rights to create index. Default is the same as used for conn
`password`	The password. Default is the same as used for conn.
`conn`	A ml.conn-class with a connection to a MarkLoic server. Optional.

Details

The function only creates and updates range index on a XML element or JSON property based on the ml.data.frame field. Information about the field can be shown by `mlDataFrame$itemField`, where mlDataFrame is a ml.data.frame object and itemField is the name of the field. Indexes created with this function will always have range-value-positions equal true.

Value

The function will raise a error if something goes wrong.

`ml.arules` *Mining Association rules and Frequent Itemsets*

Description

Mine frequent itemsets or association rules using MarkLogic Server built in Range Index functions. The function require that there is a Range Index on the underlying field of itemField, a range indexe can be created with the ml.add.index function. It will return a object that is of class rules or itemsets as defined in the arules package. It will need the arules package installed.

Usage

```
ml.arules(data, itemField, support = 0.5, confidence = 0.8, maxlen = 5,
  target = "rules")
```

Arguments

`data`	an ml.data.frame object
`itemField`	a ml.data.frame field which is the field that the itemsets will be created of. The underlying field needs to have a Range Index defined.
`support`	a numeric value for the minimal support of an item set (default: 0.5)
`confidence`	a numeric value for the minimal confidence of rules/association hyperedges (default: 0.8)
`maxlen`	an integer value for the maximal number of items per item set (default: 5)
`target`	a character string indicating the type of association mined. One of "frequent itemsets" or "rules", default is "rules"

Details

The frequent itemset and association rules extraction method is using the same method as the Apriori algorithm by first identify all 1-n itemsets that satisfy the support threshold and based on these extract rules that satisfy the confidence threshold.

It is depended on that there are a Range Index on the underlying field for the itemField. Information about the name of the field can be shown by `mlDataFrame$itemField`, where mlDataFrame is a ml.data.frame object and itemField is the name of the field.

Value

Returns an object of class rules or itemsets.

`ml.clear.database` *Remove all rfml internal files in a MarkLogic database.*

Description

The function removes the REST extensions and modules added with the ml.init.database function. It also removes the document, /rfml/rfmlInfo.json, that stores the version of the rfml package and the date the database are initiated.

Usage

```
ml.clear.database(host = "localhost", port = "8000", adminuser = "admin",
  password = "admin")
```

Arguments

`host`	The hostname or ipadress of the MarkLogic http server. Default to localhost.
`port`	The port number of the MarkLogic http server. 8000 is used default
`adminuser`	The username of a user that have rights to install options. admin is default.
`password`	The password admin is default.

Details

The user that is used for the login must have the rest-admin role, or the following privileges:

- http://marklogic.com/xdmp/privileges/rest-admin
- http://marklogic.com/xdmp/privileges/rest-writer
- http://marklogic.com/xdmp/privileges/rest-reader

Value

Nothing if success otherwise it will raise an error.

Examples

```
## Not run:
ml.clear.database("localhost", "8000", "admin", "admin")

## End(Not run)
```

ml.col.def-class *An S4 class to represent a ml.col.def.*

Description

An S4 class to represent a ml.col.def.

Slots

`.expr` A string with expresion that define the ml.col.def

`.parent` Pointer to the ml.data.frame-class object that the field belongs to

`.type` A string with the type of field

`.name` A string with name of the field

`.data_type` A string with the data type of the field

`.org_name` A string with the original names of field

`.format` A string with the format of the source field

`.xmlns` A string with the namespace of the source field

`.aggType` A string

ml.collection.info *Retrives information about a collection*

Description

The function extracts the structure of the documents belonging to a collection based on a sample it also estimates the number of documents that belongs to the collection.

Usage

```
ml.collection.info(conn, collection)
```

Arguments

`conn`	A ml.conn-class object created by ml.connect
`collection`	A string woth the name of the collection

Examples

```
## Not run:
 library(rfml)
 localConn <- ml.connect()
 ml.collection.info(localConn, "iris")

## End(Not run)
```

ml.collections *Lists all collections in a MarkLogic Database.*

Description

Lists all collections in a MarkLogic Database.

Usage

```
ml.collections(conn, query = "")
```

Arguments

conn	A ml.conn-class object created by ml.connect
query	Limit the collections based on a query. For more information about syntax see ml.data.frame

Examples

```
## Not run:
 library(rfml)
 localConn <- ml.connect()
 ml.collections(localConn)

## End(Not run)
```

ml.conn-class *An S4 class to represent a connection to a MarkLogic Server Database*

Description

An S4 class to represent a connection to a MarkLogic Server Database

Slots

`.id` A integer with the connection number.

`.host` A string with the MarkLogic Server hostname or ip-adress

`.port` A string with the port number to the HTTP server for the MarkLogic Databse used

`.mlversion` A string with the version of the MarkLogic Server

`.username` A string with username

`.password` Encrypted password

`ml.connect` *Creates a connection to a MarkLogic REST server.*

Description

Creates a connection to a MarkLogic REST server.

Usage

```
ml.connect(host = "localhost", port = "8000", username = "admin",
  password = "admin")
```

Arguments

`host`	Hostname or ip-adress of the MarkLogic http server. Default to localhost.
`port`	Port number of the MarkLogic http server. 8000 is used default
`username`	Username. admin is default.
`password`	Password admin is default.

Value

A ml.conn object.

Examples

```
## Not run:
library(rfml)
locConn <- ml.connect("localhost","8000", "admin", "admin")

## End(Not run)
```

`ml.data.frame` *Creates a ml.data.frame object*

Description

A ml.data.frame object is an abstraction layer of data stored in a MarkLogic Server database. It is created based on the provided query, collection, directory and/or fiedlFilter parameters. For query and fieldFilter parameters see details section. It present data in MarkLogic Server in a tabular format. The ml.data.frame object enables many of the operations that can be used with a data.frame object.

Usage

```
ml.data.frame(conn, query = "", fieldFilter = "", ns = "NA",
  collection = c(), directory = c(), relevanceScores = FALSE,
  docUri = FALSE)
```

Arguments

conn	A ml.conn-class object created by ml.connect
query	The query string used to define the result, see details for more information about syntax.
fieldFilter	Field level filtering. Multiple field filters are separated by , See details for limitations.
ns	A character with the namespace URI to be used with fieldFilter, default is none
collection	A list of collection URI:s to filter on.
directory	A list of directory URI:s to filter on.
relevanceScores	TRUE/FALSE. If the result attributes score, confidence and fitness should be included. Default is FALSE
docUri	TRUE/FALSE. If the uri of the documents in the results should be included. Default is FALSE.

Details

The query parameter are using the string search grammar for searching for data, all of the syntax is supported except contstraints. This enables searches such as "dog AND cat" or "dog NEAR cat". The search is always done on all fields in the data, for a more precise search use the fieldFilter.

fieldFilter enables filtering on a specific field using comparison operators can be used. For the ">" "<" "!=" "<=" ">=" operators there muset exist a element range index on the source field or a error will be raised, element range index can be created using the ml.add.index function. "==" operator will always work since it does not depend of range indexes.

Value

A ml.data.frame object.

See Also

as.data.frame for pulling data, as.ml.data.frame for uploading data, rm.ml.data.frame for delete uploaded data

Examples

```
## Not run:
 library(rfml)
 localConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(localConn, "setosa")
```

```
 # using search and collection filtering
 mlIris <- ml.data.frame(localConn, "setosa", collection = "iris")
 # using field filter
 mlIris <- ml.data.frame(localConn, fieldFilter = "Species == setosa")

## End(Not run)
```

`ml.data.frame-class` *An S4 class to represent a ml.data.frame.*

Description

An S4 class to represent a ml.data.frame.

Slots

`.name` A string with the internal name for the ml.data.frame

`.conn` The ml.conn-class object that was created with ml.connect

`.queryArgs` A list with parameters used to query MarkLogic Server

`.start` A integer with the index of the first result to get

`.nrows` A integer with the number of rows in the result

`.extracted` A logical value indicating if we have selected a subset of fields

`.col.name` A character vector with the field names

`.col.data_type` A character vector with the data types of the fields

`.col.org_name` A character vector with the original names of fields in the source documents

`.col.org_xpath` A character vector with the xpath to the original names in the source documents

`.col.format` A character vector withthe source document format XML/JSON

`.col.xmlns` A character vector with the namespace for the source document

`.col.defs` A list of ml.col.def-class added fields

`ml.init.database` *Set up a MarkLogic database for use with rfml.*

Description

The function installs REST extensions and modules needed to use the package againts a MarkLogic Server database. The function needs to be executed once for each database that is going to be used with rfml. It also creates a document, /rfml/rfmlInfo.json, that stores the version of the rfml package and the date the database are initiated.

Usage

```
ml.init.database(host = "localhost", port = "8000", adminuser = "admin",
  password = "admin")
```

Arguments

host	The hostname or ip-adress of the MarkLogic http server. Default to localhost.
port	The port number of the MarkLogic http server. 8000 is used default
adminuser	The username of a user that have rights to install options. admin is default.
password	The password admin is default.

Details

The database must have a REST server and a module database. It also adds a document, /rfml/rfmlInfo.json, that stores the version of the rfml package and the date the database are initiated.

The user that is used for the function need to have the rest-admin role, or at least the following privileges:

- http://marklogic.com/xdmp/privileges/rest-admin
- http://marklogic.com/xdmp/privileges/rest-writer
- http://marklogic.com/xdmp/privileges/rest-reader

Value

Nothing if success or raise a error.

Examples

```
## Not run:
ml.init.database("localhost", "8000", "admin", "admin")

## End(Not run)
```

---

ml.lm *Creates a simnple linear model*

Description

Returns a simple linear regression model, a linear regression model with a single explanatory variable

Usage

```
ml.lm(form, mlDf)
```

Arguments

form	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
mlDf	an ml.data.frame object

Details

The function eliminates all pairs for which either the first field or the second field is empty. After the elimination, if the length of the input is less than 2, the function returns the empty sequence. After the elimination, if the standard deviation of the independent variable is 0, the function returns a linear model with intercept = the mean of the dependent variable, coefficients = NaN and r-squared = NaN. After the elimination, if the standard deviation of the dependent variable is 0, the function returns a linear model with r-squared = NaN.

---

ml.load.sample.data *Load sample data set into MarkLogic server*

Description

The function uploads a sample data set to MarkLogic Server and returns a ml.data.frame object. Provided data sets are:

- "baskets" - sample order documents that can be used with the ml.arules function.

To remove the sample use the rm.ml.data.frame on the returned ml.data.frame object.

Usage

```
ml.load.sample.data(conn, dataSet = "baskets", name = "")
```

Arguments

conn	A ml.conn-class with a connection to a MarkLoic server
dataSet	Which dataset to upload, "baskets"
name	The name of the object. The data will be added to a collection with that name. If not provided the dataSet name is used.

Value

A ml.data.frame object pointing to the uploaded dataset.

Examples

```
## Not run:
 locConn <- ml.connect()
 mlBaskets <- ml.load.sample.data(locConn, "baskets")

## End(Not run)
```

`names,ml.data.frame-method`

Shows field names of a ml.data.frame object

Description

Shows field names of a ml.data.frame object

Usage

```
## S4 method for signature 'ml.data.frame'
names(x)
```

Arguments

x	an ml.data.frame object

`print,ml.col.def-method`

Prints information of a ml.col.def-class object.

Description

Prints information of a ml.col.def-class object.

Usage

```
## S4 method for signature 'ml.col.def'
print(x)
```

Arguments

x	an ml.col.def-class object

print,ml.data.frame-method

Prints information of a ml.data.frame object

Description

Prints information of a ml.data.frame object

Usage

```
## S4 method for signature 'ml.data.frame'
print(x)
```

Arguments

x	an ml.data.frame object

print.mlLm

Prints information for a simnple linear model returned by ml.lm

Description

Prints information for a simnple linear model returned by ml.lm

Usage

```
## S3 method for class 'mlLm'
print(x, ...)
```

Arguments

x	a ml.lm result
...	not used

`radians` *Radians*

Description

Returns numeric expression converted from degrees to radians. The function is applied when the result is returned to the client.

Usage

```
radians(x)
```

Arguments

`x`	an ml.col.def-class object.

Value

numeric expression converted from degrees to radians.

`rfml` *rfml: a R wrapper for MarkLogic REST api*

Description

rfml: a R wrapper for MarkLogic REST api

`rm.ml.data.frame` *Remove the data of a ml.data.frame object in MarkLogic server database.*

Description

Removes the data that whas saved to MarkLogic server database using the as.ml.data.frame function. If using a directory parameter it that call the same value needs to be provided for this function. The function will also delete the x object form tne R environment.

Usage

```
rm.ml.data.frame(x, directory = "")
```

Arguments

`x`	a ml.data.frame object.
`directory`	Optional. The directory where the data is stored, needs to start and end with a /.

Value

A ml.data.frame object.

See Also

`ml.data.frame`, `as.ml.data.frame` for uploading data, `as.data.frame` for pulling data

Examples

```
## Not run:
 rm.ml.data.frame(mlIris)

## End(Not run)
```

`sd,ml.col.def-method` *Standard Deviation*

Description

Returns the sample standard deviation of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
sd(x, na.rm = NULL)
```

Arguments

`x`	a ml.data.frame field.
`na.rm`	not used currently

Details

The function returns a empty value if the number of rows of the ml.data.frame that x belongs to is less than 2.

Value

The sample standard deviation

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # standard deviation
 sd(mlIris$Sepal.Length)

## End(Not run)
```

`sd.pop` *Standard Deviation of a population*

Description

Returns the sample standard deviation of a population.

Usage

```
sd.pop(x)
```

Arguments

`x`	a ml.data.frame field.

Value

The sample standard deviation of a population.

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # standard deviation
 sd.pop(mlIris$Sepal.Length)

## End(Not run)
```

---

show,ml.col.def-method

Prints information of a ml.col.def-class

Description

Prints information of a ml.col.def-class

Usage

```
## S4 method for signature 'ml.col.def'
show(object)
```

Arguments

object	an ml.col.def-class object

---

show,ml.data.frame-method

Prints information of a ml.data.frame object

Description

Prints information of a ml.data.frame object

Usage

```
## S4 method for signature 'ml.data.frame'
show(object)
```

Arguments

object	an ml.data.frame object

sum,ml.col.def-method *Sum*

Description

Returns the sum of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
sum(x, na.rm = FALSE)
```

Arguments

x	a ml.data.frame field.
na.rm	not currently used.

Value

The sum

Examples

```
## Not run:
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # sum
 sum(mlIris$Sepal.Length)

## End(Not run)
```

summary,ml.data.frame-method *ml.data.frame Summaries*

Description

ml.data.frame Summaries

Usage

```
## S4 method for signature 'ml.data.frame'
summary(object, digits = max(3L, getOption("digits")
  - 3L), maxsum = 7L, ...)
```

Arguments

object	an ml.data.frame object
digits	integer, used for number formatting
maxsum	not used.
...	not used.

---

var,ml.col.def-method *Variance*

Description

Returns the sample variance of a ml.data.frame field.

Usage

```
## S4 method for signature 'ml.col.def'
var(x, na.rm = FALSE)
```

Arguments

x	a ml.data.frame field.
na.rm	not used currently

Details

The function returns a empty value if the number of rows of the ml.data.frame that x belongs to is less than 2.

Value

The sample variance

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # return the variance
 var(mlIris$Sepal.Length)

## End(Not run)
```

var.pop *Population variance*

Description

Returns the population variance of of a ml.data.frame field.

Usage

```
var.pop(x, na.rm = FALSE)
```

Arguments

x	a ml.data.frame field.
na.rm	not used currently

Details

The function returns a empty value if the number of rows of the ml.data.frame that x belongs to is less than 2.

Value

The population variance

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on a search
 mlIris <- ml.data.frame(locConn, collection = "iris")
 # population variance
 var.pop(mlIris$Sepal.Length)

## End(Not run)
```

[,ml.data.frame-method *Extract subsets of a ml.data.frame*

Description

Extract subset of columns and/or rows of a ml.data.frame. When extracting rows a ml.col.def referense can be used or a search text, see ml.data.frame for query string grammar. See details for limitations when using a reference. The row filtering will be used togheter with the existing query of the ml.data.frame

Usage

```
## S4 method for signature 'ml.data.frame'
x[i, j, ..., drop = NA]
```

Arguments

x	a ml.data.frame from which to extract element(s).
i, j	Indices specifying elements to extract. Indices are 'numeric' or 'character' vectors or empty (missing) or 'NULL'.
...	Not used.
drop	Not used.

Details

When extracting rows using ml.col.def comparison operators can be used. For the ">" "<" "!=" "<=" ">=" operators there muset exist a element range index on the source field or a error will be raised, element range index can be created using the ml.add.index function. "==" operator will always work since it does not depend of range indexes.

Value

A ml.data.frame object is returned

Examples

```
## Not run:
 library(rfml)
 localConn <- ml.connect()
 # create a ml.data.frame based on the iris data set
 mlIris <- as.ml.data.frame(localConn, iris, "iris")
 # select first three columns
 mlIris2 <- mlIris[1:3]
 # same
 mlIris2 <- mlIris[,1:3]
 # same
 mlIris2 <- mlIris[,c("Sepal.Length","Sepal.Width","Petal.Length")]
 # select first three columns for all rows with Spieces = setosa
 mlIris2 <- mlIris[mlIris$Species=="setosa", 1:3]
 # select all columns for all rows with Spieces = setosa
 mlIris2 <- mlIris[mlIris$Species=="setosa",]
 # select all columns for all rows with "setosa" in any column
 mlIris2 <- mlIris["setosa",]

## End(Not run)
```

`$,ml.data.frame-method`

Returns a ml.data.frame field as a ml.col.def-class

Description

Returns a ml.data.frame field as a ml.col.def-class

Usage

```
## S4 method for signature 'ml.data.frame'
x$name
```

Arguments

`x`	an ml.data.frame object
`name`	field name

Value

ml.col.def-class object

`$<-,ml.data.frame-method`

Adds a new ml.data.frame field as a ml.col.def-class

Description

The fields only exists within the object and are not created at the database side.

Usage

```
## S4 replacement method for signature 'ml.data.frame'
x$name <- value
```

Arguments

`x`	A ml.data.frame object
`name`	Name of the new field
`value`	The value for the new field. Typical a expression

Value

ml.col.def-class object

Examples

```
## Not run:
 library(rfml)
 locConn <- ml.connect()
 # create a ml.data.frame based on the iris data set
 mlIris <- as.ml.data.frame(locConn, iris, "iris")
 # create a field based on an existing
 mlIris$newField <- mlIris$Petal.Width
 # create a field based calculation on existing
 mlIris$newField2 <- mlIris$Petal.Width + mlIris$Petal.Length
 # create a field based on an previous created
 mlIris$newField3 <- mlIris$Petal.Width + 10
 mlIris$abs_width <- abs(mlIris$Petal.Width)

## End(Not run)
```

Index