Package 'AdapteR'

February 21, 2016

Title This package wraps around Fuzzy Logix's DBLytix functions

Version 2.0

Description The DB Lytix(TM) suite of functions offers scalable and robust high performance analytical methods that are embedded seamlessly into database systems. Package RLytix makes it possible to use this functionality from R by providing wrapper functions around the SQL interface of DB Lytix(TM).

Depends R (>= 3.2.0),
plyr,

Imports MASS (>= 7.3-10),
Matrix (>= 1.1-5),
base,
stats

License GPL-2

LazyData true

Collate 'FLDims.R'
'FLIs.R'
'FLPrint.R'
'FLStore.R'
'utilities.R'
'FLMatrix.R'
'FLTable.R'
'FLVector.R'
'FLCastFunctions.R'
'FLCbind.R'
'FLCholeskyDecomp.R'
'FLColMeans.R'
'FLColSums.R'
'FLCorrel.R'
'FLDet.R'
'FLDiag.R'
'FLEigen.R'
'FLGinv.R'
'FLHessenDecomp.R'
'FLIdentical.R'
'FLJordanDecomp.R'
'FLKMeans.R'
'FLLUDecomp.R'
'FLLength.R'

'data_prep.R'
'FLLinRegr.R'
'FLLogRegr.R'
'FLMatrixArithmetic.R'
'FLconstructSQL.R'
'FLMatrixBind.R'
'FLMatrixClasses.R'
'FLMatrixNorm.R'
'FLMatrixREF.R'
'FLMatrixRREF.R'
'FLQRDecomp.R'
'FLRankMatrix.R'
'FLRbind.R'
'FLRowMeans.R'
'FLRowSums.R'
'FLSV.R'
'FLSVDecomp.R'
'FLSolve.R'
'FLSolveExcl.R'
'FLSubsetting.R'
'FLTrace.R'
'FLTranspose.R'
'FLTriDiag.R'

RoxygenNote 5.0.1

R topics documented:

* *Element-Wise Multiplication of in-database objects.*

Description

* does the Element-wise Multiplication of in-database objects.

Usage

```
"*"(pObj1, pObj2)
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Element-wise Multiplication of in-database objects mimics the normal Element-wise Multiplication of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

* returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
ResultFLmatrix <- flmatrix * Rvector
```

+ *Addition of in-database objects.*

Description

+ does the addition of in-database objects.

Usage

```
"+"(pObj1, pObj2)
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The addition of in-database objects mimics the normal addition of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

+ returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
ResultFLmatrix <- flmatrix + Rvector
```

/ *Element-wise Division of in-database objects.*

Description

/ does the Element-wise Division of in-database objects.

Usage

```
"/"(pObj1, pObj2)
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Element-wise Division of in-database objects mimics the / of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

/ returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Constraints

division by 0 gives inf in R, but is not supported for in-database objects

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
ResultFLmatrix <- flmatrix / Rvector
```

==.FLMatrix *Equality of in-database objects.*

Description

== checks the equality of in-database objects.

Usage

```
## S3 method for class 'FLMatrix'
pObj1 == pObj2
```

Arguments

x	can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The equality of in-database objects mimics the normal addition of R data types. One can check equality of FLMatrices, FLMatrix - R matrices, FLVectors and FLVector - RVector.

Value

== returns a logical TRUE or FALSE.

Constraints

Currently only dgCMatrix,dgeMatrix,dsCMatrix, dgTMatrix,matrix,Matrix,vector R types are supported. Comparision of FLMatrix with FLVector is not currently Supported.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
Result <- flmatrix == flmatrix
Result <- Rvector == as.FLVector(Rvector,connection)
```

as.FLMatrix.Matrix *Casting to FLMatrix*

Description

Converts input m to FLMatrix object In addition, one can specify number of rows and columns of resulting flmatrix object

Usage

```
as.FLMatrix.Matrix(object, connection, ...)
```

Arguments

object	matrix, vector, data frame, sparseMatrix, FLVector which needs to be casted to FLMatrix
connection	ODBC connection object
...	
sparse	
nr	number of rows in resulting FLMatrix
nc	number of columns in resulting FLMatrix. nr and nc inputs are applicable only in case of vector,FLVector

Value

FLMatrix object after casting.

as.FLVector *casting to FLVector*

Description

Converts input obj to FLVector object

Usage

```
as.FLVector(object, connection, ...)
```

Arguments

connection	ODBC connection object
obj	matrix, vector, data frame, sparseMatrix, FLMatrix which needs to be casted to FLVector
size	number of elements in resulting FLVector. size input is not applicable only in case of FLMatrix

Value

FLVector object after casting.

as.matrix *Converts* x *to matrix in R*

Description

Converts x to matrix in R

Usage

```
as.matrix(x, ...)
```

as.matrix.FLVector *Converts FLVector object to a matrix in R*

Description

Converts FLVector object to a matrix in R

Usage

```
## S3 method for class 'FLVector'
as.matrix(obj)
```

as.vector.FLMatrix *Converts FLMatrix object to vector in R*

Description

Converts FLMatrix object to vector in R

Usage

```
as.vector.FLMatrix(object, mode = "any")
```

as.vector.FLVector *Converts FLVector object to vector in R*

Description

Converts FLVector object to vector in R

Usage

```
as.vector.FLVector(object, mode = "any")
```

cbind *Combine objects by columns.*

Description

cbind combines input objects by columns and forms a FLMatrix.

Usage

```
cbind(x, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

cbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by columns and makes a FLMatrix.

Value

cbind returns a FLMatrix object which is the column-wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of rows.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- cbind(flmatrix,1:5,flmatrix)
```

cbind.FLMatrix *Combine objects by columns.*

Description

cbind combines input objects by columns and forms a FLMatrix.

Usage

```
## S3 method for class 'FLMatrix'
cbind(object, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

cbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by columns and makes a FLMatrix.

Value

cbind returns a FLMatrix object which is the column-wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of rows.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- cbind(flmatrix,1:5,flmatrix)
```

---

chol.FLMatrix *Cholesky Decomposition.*

Description

chol computes the Cholesky factorization of FLMatrix object.
The Cholesky decomposition is a decomposition of a positive definite matrix into the product of a lower triangular matrix and its conjugate transpose.

Usage

```
## S3 method for class 'FLMatrix'
chol(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads chol and implicitly calls FLCholeskyDecompUdt.

Value

chol returns FLMatrix which is the upper triangular factor of the Cholesky decomposition

Constraints

Input can only be a Hermitian, positive definite square matrix (n x n) with maximum dimension limitations of (1000 x 1000)

Examples

```
connection<-odbcConnect("Gandalf")
flmatrix<-FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- chol(flmatrix)
```

---

colMeans.FLMatrix *column means of a FLMatrix.*

Description

colMeans computes the column-wise average of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
colMeans(object)
```

Arguments

object	is of class FLMatrix.

Details

The wrapper overloads colMeans and extends it to FLMatrix objects.

Value

colMeans returns a FLVector object representing the column-wise Means.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLVector <- colMeans(flmatrix)
```

---

colSums.FLMatrix *column sums of a FLMatrix.*

Description

colSums computes the column-wise sums of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
colSums(object)
```

Arguments

object	is of class FLMatrix.

Details

The wrapper overloads colSums and extends it to FLMatrix objects.

Value

`colSums` returns a FLVector object representing the col-wise sums.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLVector <- colSums(flmatrix)
```

`constructSelect` *constructs a sql statement returning the deep table representation of the object.*

Description

constructs a sql statement returning the deep table representation of the object.

Usage

```
constructSelect(object, ...)
```

Arguments

`object`	the object to query
`...`	arguments passed on to SQL generation. see joinNames

Value

a character SQL representation

Author(s)

Gregor Kappler <g.kappler@gmx.net>

cor.FLMatrix *Correlation.*

Description

cor computes correlation of FLVectors: x and y.

Usage

```
## S3 method for class 'FLVector'
cor(x, y = x)
```

Arguments

x	A numeric vector,matrix or data frame
y	A vector,matrix or data frame with compatible dimensions to x

Details

The wrapper overloads cor and implicitly calls FLCorrel.

Value

cor returns correlation of x and y.

Constraints

The number of non-null pairs must be greater than or equal to 2. If number of non-null pairs is less than 2, FLCorrel returns a NULL.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
table <- FLTable(connection, "FL_TRAIN", "tblAbaloneWide", "ObsID")
cor(table,table)
```

det.FLMatrix *Determinant of a Matrix.*

Description

det computes the determinant of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
det(object)
```

Arguments

object	is a FLMatrix object

Details

The wrapper overloads det and implicitly calls FLMatrixDetUdt.

Value

det returns determinant stored in-database as FLVector which replicates the equivalent R vector output.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLvector <- det(flmatrix)
```

---

diag *Matrix Diagonals*

Description

Extract or replace the diagonal of a matrix, or construct a diagonal matrix.

Usage

```
diag(x, ...)
```

Arguments

x	is an object of class FLMatrix or FLVector

Details

diag has three distinct usages: x is a FLMatrix, when it extracts the diagonal. x is a scalar (length-one FLVector) and the only argument, it returns a square identity matrix of size given by the scalar. x is a FLVector, either of length at least 2. This returns a square matrix with the given diagonal entries.

Value

If x is a FLMatrix then diag(x) returns the diagonal of x as FLVector object. If x is FLVector, the value is a diagonal square FLMatrix with diagonal elements as given in FLVector.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLVector <- diag(flmatrix)
DeepTable <- FLTable(connection, "FL_TRAIN", "tblVectorDeep","vector_id","vector_key","vector_value")
flvectorDeep <- FLVector(DeepTable,"vector_value",1)
resultFLMatrix <- diag(flvectorDeep)
```

---

eigen.FLMatrix *Spectral Decomposition of a Matrix.*

Description

eigen Computes eigenvalues and eigenvectors of FLMatrices.

Usage

```
## S3 method for class 'FLMatrix'
eigen(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads eigen and implicitly calls FLEigenValueUdt and FLEigenVectorUdt.

Value

eigen returns a list of FLMatrix object containing the eigen vectors and a FLVector object containing eigen values which replicates the equivalent R output.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultList <- eigen(flmatrix)
resultList$valueColumns
resultList$vectors
```

FLamendDimnames *Amends dimension names to a matrix if a mapping exists.*

Description

If no mapping exists uses the unique row and column names. Dimnames 1:n are set to NULL in order to adhere to R conventions.

Usage

```
FLamendDimnames(flm, map_table)
```

Arguments

flm	FLMatrix

Value

the FLMatrix object, with slot dimnames re set

Author(s)

Gregor Kappler <g.kappler@gmx.net>

FLHessen.FLMatrix *Hessenberg Decomposition of a Matrix.*

Description

FLHessen computes the Hessenberg decomposition for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
FLHessen(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads hessen and implicitly calls FLHessenbergDecompUdt.

Value

FLHessen returns a list of two components:

P	FLMatrix representing P matrix obtained from Hessenberg decomposition
H	FLMatrix representing H matrix obtained from Hessenberg decomposition

Constraints

Input can only be square matrix with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultList <- FLHessen(flmatrix)
resultList$P
resultList$H
```

FLJordan.FLMatrix *Jordan Decomposition of a Matrix.*

Description

jordan computes the Jordan decomposition for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
FLJordan(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads jordan and implicitly calls FLJordanDecompUdt.

Value

jordan returns a list of two components:

J	FLVector representing J vector obtained from Jordan decomposition
P	FLMatrix representing P matrix obtained from Jordan decomposition
PInv	FLMatrix representing PInv matrix obtained from Jordan decomposition

Constraints

Input can only be square matrix with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultList <- jordan(flmatrix)
resultList$J
resultList$P
resultList$PInv
```

FLKMeans-class *An S4 class to represent FLKMeans*

Description

An S4 class to represent FLKMeans

Arguments

object	retrieves the cluster vector
object	retrieves the coordinates of the centroids
object	overloads the print function
object	total within sum of squares
object	within sum of squares
object	between sum of squares
object	total sum of squares
object	size vector

Slots

`no_of_centers` A numeric vector containing the number of clusters, say k

`AnalysisID` A character output used to retrieve the results of analysis

`connection` ODBC connectivity for R

`table` FLTable object given as input on which analysis is performed

`resultsfetched` A logical vector describing what components are fetched

`results` A list of all fetched components

`deeptablename` A character vector containing a deeptable(either conversion from a widetable or input deeptable)

FLLinRegr-class *An S4 class to represent FLLinRegr*

Description

An S4 class to represent FLLinRegr

Arguments

object	contains: call,coefficients
object	a named vector of coefficients
object	contains: call,residuals,coefficients,significant codes note and statistical output.

Slots

`formula` an object of class 'formula': Model Formulae

`table_name` A character

`deeptablename` A character vector containing name of the deeptable on conversion from a widetable

`AnalysisID` An output character ID from CALL FLLinRegr

`dataprepID` An output character ID from CALL FLRegrDataPrep

`datatable` An object of class FLTable

`FLLogRegr-class` *An S4 class to represent FLLogRegr*

Description

An S4 class to represent FLLogRegr

Arguments

`object`	contains: call,coefficients
`object`	a named vector of coefficients
`object`	contains: call,residuals,coefficients,significant codes note and statistical output.

Slots

`formula` an object of class 'formula': Model Formulae

`table_name` A character

`deeptablename` A character vector containing name of the deeptable on conversion from a widetable

`AnalysisID` An output character ID from CALL FLLogRegr

`dataprepID` An output character ID from CALL FLRegrDataPrep

`datatable` An object of class FLTable

`FLLU-class` *An S4 class to represent LU Decomposition*

Description

An S4 class to represent LU Decomposition

Slots

`x` object of class FLVector

`perm` object of class FLVector

`Dim` object of class FLVector

`lower` object of class FLMatrix

`upper` object of class FLMatrix

`data_perm` object of class FLMatrix

FLMatrix *Constructor function for FLMatrix.*

Description

FLMatrix constructs an object of class FLMatrix.

Usage

```
FLMatrix(connection, database = getOption("ResultDatabaseFL"), table_name,
  matrix_id_value = "", matrix_id_colname = "",
  row_id_colname = "rowIdColumn", col_id_colname = "colIdColumn",
  cell_val_colname = "valueColumn", dim = 0, dimnames = NULL,
  conditionDims = c(FALSE, FALSE), whereconditions = c(""),
  map_table = NULL)
```

Arguments

connection	ODBC connection handle as returned by odbcConnect
database	name of the database
table_name	name of the matrix table
matrix_id_value	identifier for the input matrix
matrix_id_colname	matrix id value in table_name
row_id_colname	column name in table_name where row numbers are stored
col_id_colname	column name in table_name where column numbers are stored
cell_val_colname	column name in table_name where matrix elements are stored

Details

FLMatrix object is an in-database equivalent to matrix object. This object is used as input for matrix operation functions.

Value

FLMatrix returns an object of class FLMatrix mapped to an in-database matrix.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
```

FLMatrix-class *An S4 class to represent FLMatrix. A Matrix can be either based off a query from a deep table (customizable by any where-condition) – or based off an arbitrary SQL statement returning a deep table.*

Description

An S4 class to represent FLMatrix. A Matrix can be either based off a query from a deep table (customizable by any where-condition) – or based off an arbitrary SQL statement returning a deep table.

Slots

`dimnames` list of 2 elements with row, column names of FLMatrix object

`dim` list of 2 FLTableQuery instances (or NULL) that map row_ids in the select to row-names in R

FLMatrixArithmetic *Arithmetic operations on in-database objects.*

Description

- does the subtraction of in-database objects.

Usage

```
FLMatrixArithmetic(pObj1, pObj2, pOperator)
```

Details

All the operators work the same way with in-database objects as they do with R data types. The result is an in-database object.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
ResultFLmatrix <- flmatrix - Rvector
```

FLMatrixBind *Bind a matrix/array by an index. Currently limited to matrices with character dimnames*

Description

Bind a matrix/array by an index. Currently limited to matrices with character dimnames

Usage

```
FLMatrixBind(parts, by)
```

Arguments

parts	
by	the numeric index by which binding takes place

Value

returns a remote matrix object defining the deep table sql for the *bound result.

Author(s)

Gregor Kappler <g.kappler@gmx.net>

FLMatrixNorm.FLMatrix *Norm of a Matrix.*

Description

FLMatrixNorm gives the value of Norm for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
FLMatrixNorm(object, NormMethod)
```

Arguments

object	is of class FLMatrix
NormMethod	is an integer from 1-4 representing the type of norm that should be computed.

Details

The wrapper overloads FLMatrixNorm and implicitly calls FLMatrixNormUdt.

Value

FLMatrixNorm returns a FLVector object which is the Norm of input FLMatrix object calculated using method specified by NormMethod input. There are 4 types of norms of a matrix:

1-Norm	Maximum of the sum of the absolute values for the columns
2-Norm	Maximum of the sum of the absolute values for the rows
Frobenius Norm	Square root of the trace of (t(A)A)
Infinity Norm	Square root of the maximum of the magnitudes of the Eigenvalues of (t(A)A)

Constraints

Input can only be with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLVector <- FLMatrixNorm(flmatrix,4)
```

FLMatrixREF.FLMatrix *Row Echelon form of a Matrix.*

Description

FLMatrixREF gives the Row Echelon form of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
FLMatrixREF(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads FLMatrixREF and implicitly calls FLMatrixREFUdt.

Value

FLMatrixREF returns a FLMatrix object which is the Row Echelon form of input FLMatrix.

Constraints

Input can only be a square FLMatrix with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- FLMatrixREF(flmatrix)
```

FLMatrixRREF.FLMatrix *Reduced Row Echelon form of a Matrix.*

Description

FLMatrixRREF gives the Reduced Row Echelon form of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
FLMatrixRREF(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads FLMatrixRREF and implicitly calls FLMatrixRREFUdt.

Value

FLMatrixRREF returns a FLMatrix object which is the Reduced Row Echelon form of input FLMatrix.

Constraints

Input can only be a square FLMatrix with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- FLMatrixRREF(flmatrix)
```

FLSelectFrom-class *A selectFrom models a select from a table*

Description

A selectFrom models a select from a table

Slots

database character

table_name character

FLSolveExcl.FLMatrix *Inverse of a Matrix excluding a dimension.*

Description

solveExcl computes the inverse for FLMatrix objects by excluding the specified row and column from the matrix.

Usage

```
## S3 method for class 'FLMatrix'
FLSolveExcl(object, ExclIdx)
```

Arguments

object	is of class FLMatrix
ExclIdx	is a positive integer specifying row or column id to be excluded.

Details

The wrapper overloads solveExcl and implicitly calls FLMatrixInvExclUdt.

Value

solveExcl returns a FLMatrix object which is the inverse of input FLMatrix object after excluding given dimension.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- solveExcl(flmatrix,3)
```

FLSV.FLMatrix *Singular Values of a FLMatrix.*

Description

FLSV computes the singular values for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
FLSV(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads FLSV and implicitly calls FLSVUdt.

Value

FLSV returns a FLVector object representing the singular values.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLVector <- FLSV(flmatrix)
```

FLTable *Constructor function for FLTable.*

Description

FLTable constructs an object of class FLTable.

Usage

```
FLTable(connection, database, table, obs_id_colname,
  var_id_colnames = character(0), cell_val_colname = character(0),
  whereconditions = character(0))
```

Arguments

connection	ODBC connection handle as returned by odbcConnect
database	name of the database
table	name of the table
obs_id_colname	column name set as primary key
cell_val_colname	column name where cell values are stored if FLTable is deep
var_id_colname	column name where variable id's are stored if FLTable is deep

Details

FLTable refers to an in-database table. This is equivalent to data.frame object. This object is commonly used as input for data mining functions.

Value

FLTable returns an object of class FLTable mapped to a table in Teradata.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_TRAIN", "tblAbaloneWide", "ObsID")
names(widetable)
```

FLTable-class *An S4 class to represent FLTable*

Description

An S4 class to represent FLTable

Arguments

object	retrieves the column names of FLTable object

Slots

odbc_connection ODBC connectivity for R

db_name character

table_name character

obs_id_colname character

var_id_colnames character

cell_val_colname character

isDeep logical

FLTableFunctionQuery-class *A TableFunctionQuery models a select from an arbitrary query*

Description

A TableFunctionQuery models a select from an arbitrary query

Slots

SQLquery character

FLTriDiag.FLMatrix *TriDiagonal or Upper Hessenberg matrix of a FLMatrix.*

Description

FLTriDiag computes the TriDiagonal or Upper Hessenberg matrix of FLMatrix object.

Usage

```
## S3 method for class 'FLMatrix'
FLTriDiag(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads FLTriDiag and implicitly calls FLTriDiagUdt.

Value

FLTriDiag returns a FLMatrix object representing the upper Hessenberg or TriDiagonal matrix.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (700 x 700).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- FLTriDiag(flmatrix)
```

FLVector *Constructor function for FLVector, representing a vector in database, either a deep or a wide matrix. gk: how can we support row vectors?*

Description

FLVector constructs an object of class FLVector.

Usage

```
FLVector(table, val_col_name = character(), val_row_name = character(),
  whereconditions = character())
```

Arguments

table	FLTable object as returned by FLTable where the vector is stored
colname	name of the column in table where vector elements are stored
vector_id_value	unique identifier for the vector if stored in deep table

Details

FLVector object is an in-database equivalent to a vector. This object is used as input for matrix and arithmetic operation functions.

Value

FLVector returns an object of class FLVector mapped to an in-database vector.

See Also

FLTable

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
WideTable <- FLTable(connection, "FL_TRAIN", "tblVectorWide","vector_key")
flvectorWide <- FLVector(WideTable,"vector_value")
DeepTable <- FLTable(connection, "FL_TRAIN", "tblVectorDeep","vector_id","vector_key","vector_value")
flvectorDeep <- FLVector(DeepTable,"vector_value",1)
```

FLVector-class *An S4 class to represent FLVector*

Description

An S4 class to represent FLVector

---

ginv.FLMatrix *Generalized Inverse of a Matrix.*

Description

ginv computes the pseudo-inverse for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
ginv(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads ginv and implicitly calls FLMatrixPseudoInvUdt.

Value

ginv returns a FLMatrix object which is the pseudo-inverse of input FLMatrix object and replicates the equivalent R output.

Constraints

Input can only be with maximum dimension limitations of (500 x 500).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
resultFLMatrix <- ginv(flmatrix)
```

---

glm.FLTable *Linear Regression.*

Description

glm performs linear regression on FLTable objects.

Usage

```
## S3 method for class 'FLTable'
glm(formula, data, rushil = "binomial", iter = 25,
  threshold = 0.1, ...)
```

Arguments

formula	A symbolic description of model to be fitted
data	An object of class FLTable

Details

The wrapper overloads glm and implicitly calls FLRegrDataPrep and FLLogRegr.

Value

glm performs linear regression and replicates equivalent R output.

Constraints

None

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_REV4546", "tblAbaloneWide", "ObsID")
glmfit <- glm(Rings~Height+Diameter,widetable)
```

`identical` *Equality of in-database objects.*

Description

`identical` checks the equality of in-database objects.

Usage

```
identical(pObj1, pObj2)
```

Arguments

`x`	can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
`y`	can be an in-database object like FLMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The equality of in-database objects mimics the normal addition of R data types. One can check equality of FLMatrices, FLMatrix - R matrices, FLVectors and FLVector - RVector.

Value

`identical` returns a logical TRUE or FALSE.

Constraints

Currently only `dgCMatrix`,`dgeMatrix`,`dsCMatrix`, `dgTMatrix`,`matrix`,`Matrix`,`vector` R types are supported.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
Result <- identical(flmatrix,flmatrix)
Result <- identical(Rvector,as.FLVector(Rvector,connection))
```

is.FLMatrix *Check if the object is an FLMatrix object*

Description

Check if the object is an FLMatrix object

Usage

```
is.FLMatrix(object)
```

is.FLTable *Check if the object is an FLTable object*

Description

Check if the object is an FLTable object

Usage

```
is.FLTable(object)
```

is.FLVector *Check if the object is an FLVector object*

Description

Check if the object is an FLVector object

Usage

```
is.FLVector(object)
```

kmeans.FLTable *K-Means Clustering.*

Description

kmeans performs k-means clustering on FLTable objects.

Usage

```
## S3 method for class 'FLTable'
kmeans(table, centers, max.iter = 10, nstart = 1,
  exclude = as.character(c()), class_spec = list(), where_clause = "")
```

Arguments

table	an object of class FLTable
centers	the number of clusters
max.iter	the maximum number of iterations allowed
nstart	the initial number of random sets
exclude	the comma separated character string of columns to be excluded
class_spec	list describing the categorical dummy variables
where_clause	takes the where_clause as a string

Details

The wrapper overloads kmeans and implicitly calls FLKMeans.

Value

kmeans performs k-means clustering and replicates equivalent R output.

Constraints

None

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_TRAIN", "tblAbaloneWide", "ObsID")
kmeansobject <- kmeans(widetable,3,20,2,"Rings,SEX",list("DummyCat(D)","SEX(M)"))
print(kmeansobject)
```

length.FLMatrix *computes the length of FLMatrix object.*

Description

computes the length of FLMatrix object.

Usage

```
## S3 method for class 'FLMatrix'
length(obj)
```

Arguments

obj	is a FLMatrix object.

Value

length returns a R Vector giving the length of input object.

length.FLTable *computes the length of FLTable object.*

Description

computes the length of FLTable object.

Usage

```
## S3 method for class 'FLTable'
length(obj)
```

Arguments

obj	is a FLTable object.

Value

length returns a R Vector giving the number of observations or rows in input FLTable object.

length.FLVector *computes the length of FLVector object.*

Description

computes the length of FLVector object.

Usage

```
## S3 method for class 'FLVector'
length(obj)
```

Arguments

obj	is a FLVector object.

Value

length returns a R Vector giving the length of input object.

lm.FLTable *Linear Regression.*

Description

lm performs linear regression on FLTable objects.

Usage

```
## S3 method for class 'FLTable'
lm(formula, data, ...)
```

Arguments

formula	A symbolic description of model to be fitted
data	An object of class FLTable

Details

The wrapper overloads lm and implicitly calls FLRegrDataPrep and FLLinRegr.

Value

lm performs linear regression and replicates equivalent R output.

Constraints

None

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
widetable  <- FLTable(connection, "FL_REV4546", "tblAbaloneWide", "ObsID")
lmfit <- lm(Rings~Height+Diameter,widetable)
```

`lu.FLMatrix` *LU Decomposition.*

Description

The LU decomposition involves factorizing a matrix as the product of a lower triangular matrix L and an upper triangular matrix U. Permutation matrix is also provided in the output. If permutation matrix is not used in the decomposition, the output of permutation matrix is an identity matrix.

Usage

```
## S3 method for class 'FLMatrix'
lu(object)
```

Arguments

`object`	is of class FLMatrix

Details

`lu` replicates the equivalent lu() generic function.
The wrapper overloads lu and implicitly calls FLLUDecompUdt.

`expand` decomposes the compact form to a list of matrix factors.
The expand method returns L,U and P factors as a list of FLMatrices.

The decomposition is of the form A = P L U where typically all matrices are of size (n x n), and the matrix P is a permutation matrix, L is lower triangular and U is upper triangular.

Value

`x`	the FLVector form of "L" (unit lower triangular) and "U" (upper triangular) factors of the original matrix
`perm`	FLVector that describes the permutation applied to the rows of the original matrix
`Dim`	FLVector that gives the dimension of the original matrix
`lower`	FLMatrix representing the lower triangular matrix
`upper`	FLMatrix representing the upper triangular matrix
`data_perm`	FLMatrix representing the permutation matrix

Constraints

Input can only be with maximum dimension limitations of (1000 x 1000).

Examples

```
connection<-odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
FLLUobject <- lu(flmatrix)
expand(FLLUobject)
expand(lu(flmatrix))$L
expand(lu(flmatrix))$U
expand(lu(flmatrix))$P
```

`names.FLTable` *Gives the column names of FLTable object*

Description

Gives the column names of FLTable object

Usage

```
## S3 method for class 'FLTable'
names(object)
```

Arguments

`object`

`qr.FLMatrix` *QR Decomposition.*

Description

The QR decomposition involves factorizing a matrix into QMatrix and RMatrix.

Usage

```
## S3 method for class 'FLMatrix'
qr(object)
```

Arguments

`object`	is of class FLMatrix

Details

`qr` replicates the equivalent qr() generic function.
The wrapper overloads qr and implicitly calls FLQRDecompUdt.

Value

qr returns a list of five components:

qr	a FLMatrix with the same dimensions as object. The upper triangle contains the R of the decomposition and the lower triangle contains information on the Q of the decomposition (stored in compact form)
qraux	a FLVector of length ncol(object) which contains additional information on Q.
rank	the FLVector giving rank of object
QMatrix	the resulting Q Matrix stored in-database as FLMatrix
RMatrix	the resulting R Matrix stored in-database as FLMatrix

Constraints

Input can only be with maximum dimension limitations of (700 x 700).

Examples

```
connection<-odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultList <- qr(flmatrix)
resultList$qr
resultList$qraux
resultList$rank
resultList$QMatrix
resultList$RMatrix
```

---

rankMatrix.FLMatrix *Matrix Rank.*

Description

rankMatrix computes the rank of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
rankMatrix(object)
```

Arguments

object	is of class FLMatrix

Details

rankMatrix computes the rank of input FLMatrix object, stores the result in-database and returns FLVector object

Value

rankMatrix returns FLVector object of size 1 which replicates the equivalent R output.

Constraints

Input can have maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLVector <- rankMatrix(flmatrix)
```

---

rbind *Combine objects by rows.*

Description

rbind combines input objects by rows and forms a FLMatrix.

Usage

```
rbind(x, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

rbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by rows and makes a FLMatrix.

Value

rbind returns a FLMatrix object which is the row-wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of columns.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- rbind(flmatrix,1:5,flmatrix)
```

rbind.FLMatrix *Combine objects by rows.*

Description

rbind combines input objects by rows and forms a FLMatrix.

Usage

```
## S3 method for class 'FLMatrix'
rbind(object, ...)
```

Arguments

x...	can be a sequence of vector, FLVector, matrix, FLMatrix or data frames

Details

rbind takes a sequence of vector, FLVector, matrix, FLMatrix or data frames arguments, combines them by rows and makes a FLMatrix.

Value

rbind returns a FLMatrix object which is the row wise combination of input arguments.

Constraints

Input matrices, FLMatrices and data frames should have same number of columns.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLMatrix <- rbind(flmatrix,1:5,flmatrix)
```

restrictFLMatrix *Appends where clauses for subsetting etc.*

Description

Appends where clauses for subsetting etc.

Usage

```
restrictFLMatrix(object, whereconditions = object@select@whereconditions,
  dimnames = object@dimnames, conditionDims = c(FALSE, FALSE))
```

Arguments

object	An FLMatrix object
whereconditions	constraints to be added
dimnames	new dimension names
conditionDims	vector of 2 LOGICAL values, if first is TRUE, a inCondition for the rownames is appended, if 2 for the columns respectively.

---

rowMeans.FLMatrix *row means of a FLMatrix.*

Description

rowMeans computes the row-wise average of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
rowMeans(object)
```

Arguments

object	is of class FLMatrix.

Details

The wrapper overloads rowMeans and extends it to FLMatrix objects.

Value

rowMeans returns a FLVector object representing the row-wise Means.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLVector <- rowMeans(flmatrix)
```

rowSums.FLMatrix *row sums of a FLMatrix.*

Description

rowSums computes the row-wise sums of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
rowSums(object)
```

Arguments

object	is of class FLMatrix.

Details

The wrapper overloads rowSums and extends it to FLMatrix objects.

Value

rowSums returns a FLVector object representing the row-wise sums.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
resultFLVector <- rowSums(flmatrix)
```

solve.FLMatrix *Inverse of a Matrix.*

Description

solve computes the inverse for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
solve(pObj1)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads solve and implicitly calls FLMatrixInvUdt.

Value

solve returns a FLMatrix object which is the inverse of input FLMatrix object and replicates the equivalent R output.

Constraints

Input can only be a square matrix (n x n) with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLMatrix <- solve(flmatrix)
```

---

store *stores a matrix in a table. TODO: define when data is stored (automatic caching, user requests...)*

Description

stores a matrix in a table. TODO: define when data is stored (automatic caching, user requests...)

Usage

```
store(object, returnType, connection, ...)
```

Arguments

object	the object to store

Value

A FLMatrix based on a stored table of the executed

Author(s)

Gregor Kappler <g.kappler@gmx.net>

svd.FLMatrix *Singular Value Decomposition of a Matrix.*

Description

svd computes the singular value decomposition for FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
svd(object, nu = c(), nv = c())
```

Arguments

object	is of class FLMatrix
nu	number of left singular vectors to be computed.This must between 0 and nrow(object).
nv	number of right singular vectors to be computed.This must between 0 and ncol(object).

Details

The wrapper overloads svd and implicitly calls FLSVDUdt.

Value

svd returns a list of three components:

d	a FLVector containing the singular values of x, of size min(n, p).
u	a FLMatrix whose columns contain the left singular vectors of x, present if nu > 0. Dimension c(n, nu).
v	a FLMatrix whose columns contain the right singular vectors of x, present if nv > 0. Dimension c(p, nv).

Constraints

Input can only be with maximum dimension limitations of (550 x 550).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 3)
resultList <- svd(flmatrix)
resultList$d
resultList$u
resultList$v
```

t.FLMatrix *Matrix Transpose.*

Description

t returns the transpose of FLMatrix objects.

Usage

```
## S3 method for class 'FLMatrix'
t(object)
```

Arguments

object	is of class FLMatrix

Details

The wrapper overloads t such that given a matrix of class FLMatrix, t returns the transpose of that object

Value

t returns a FLMatrix object which is the transpose of input FLMatrix object and replicates the equivalent R output.

Constraints

Input can be a matrix of dimensions (m x n) where m > n, m < n or m = n.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLMatrix <- t(flmatrix)
```

tr *Matrix Trace.*

Description

tr computes the trace of FLMatrix objects.

Usage

```
tr(x, ...)
```

Arguments

x	an object of class FLMatrix

Details

tr computes the trace of input FLMatrix object, stores the result in-database and returns FLVector object

Value

tr returns FLVector object of size 1 which replicates the equivalent R output.

Constraints

Input can only be with maximum dimension limitations of (1000 x 1000).

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLVector <- tr(flmatrix)
```

---

viewSelectMatrix,FLMatrix,character,character-method

todo: alias

Description

todo: alias

Usage

```
## S4 method for signature 'FLMatrix,character,character'
viewSelectMatrix(object, localName,
  withName = "z")
```

[.FLMatrix

Extract part of FLMatrix object.

Description

[] acts on FLMatrix objects and extracts parts of them.

Usage

```
## S3 method for class 'FLMatrix'
object[rows = 1, cols = 1, drop = TRUE]
```

Arguments

object	is a FLMatrix object
rows	is a vector input corresponding to rows to be extracted
cols	is a vector input corresponding to columns to be extracted

Value

[] returns FLMatrix object after extraction which replicates the equivalent R extraction.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLmatrix <- flmatrix[1,]
```

[.FLTable *Extract part of FLMatrix object.*

Description

[] acts on FLMatrix objects and extracts parts of them.

Usage

```
## S3 method for class 'FLTable'
object[rows = 1, cols = 1, drop = TRUE]
```

Arguments

object	is a FLMatrix object
rows	is a vector input corresponding to rows to be extracted
cols	is a vector input corresponding to columns to be extracted

Value

[] returns FLMatrix object after extraction which replicates the equivalent R extraction.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 2)
resultFLmatrix <- flmatrix[1,]
```

[.FLVector *Extract part of FLVector object.*

Description

[] acts on FLVector objects and extracts parts of them.

Usage

```
## S3 method for class 'FLVector'
object[pSet = 1:length(object)]
```

Arguments

pSet	is a vector representing the indices of elements to extract
pObj	is a FLVector object

Value

[] returns FLVector object after extraction which replicates the equivalent R extraction.

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
WideTable <- FLTable(connection, "FL_TRAIN", "tblVectorWide","vector_key")
flvectorWide <- FLVector(WideTable,"vector_value")
resultFLVector <- flvectorWide[1:2]
DeepTable <- FLTable(connection, "FL_TRAIN", "tblVectorDeep","vector_id","vector_key","vector_value")
flvectorDeep <- FLVector(DeepTable,"vector_value",1)
resultFLVector <- flvectorDeep[1:2]
```

%% *remainder of division on in-database objects.*

Description

%% calculates the remainder of in-database object division.

Usage

```
pObj1 %% pObj2
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The remainder of in-database objects mimics the normal remainder of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

%% returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Constraints

division by 0 gives inf in R, but is not supported for in-database objects

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
ResultFLmatrix <- flmatrix %% Rvector
```

%/% *Integer Division of in-database objects.*

Description

%/% does the Element-wise Integer Division of in-database objects.

Usage

```
pObj1 %/% pObj2
```

Arguments

x	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
y	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Element-wise Integer Division of in-database objects mimics the %/% of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

%/% returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Constraints

division by 0 gives inf in R, but is not supported for in-database objects

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 1)
Rvector <- 1:5
ResultFLmatrix <- flmatrix %/% Rvector
```

`%*%` *Cross-Product of in-database objects.*

Description

`%*%` does the Cross-Product of in-database objects.

Usage

```
pObj1 %*% pObj2
```

Arguments

`x`	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector
`y`	can be an in-database object like FLMatrix,FLSparseMatrix,FLVector or a normal R object like matrix,sparseMatrix,vector

Details

The Cross-Product of in-database objects mimics the `%*%` of R data types. All combinations of operands are possible just like in R and the result is an in-database object.

Value

`%*%` returns an in-database object if there is atleast one in-database object as input.Otherwise, the default behavior of R is preserved

Examples

```
library(RODBC)
connection <- odbcConnect("Gandalf")
flmatrix <- FLMatrix(connection, "FL_TRAIN", "tblMatrixMulti", 5)
Rvector <- 1:5
ResultFLmatrix <- flmatrix %*% Rvector
```

Index