Package ‘scidb’

February 20, 2015

Type Package
Title An R interface to SciDB
Version 1.1-2
Date 2014-03-16
Author Paradigm4, B. W. Lewis
Maintainer B. W. Lewis <blewis@illposed.net>
Copyright Paradigm4, Inc.
Description SciDB is an open-source array database (http://scidb.org). The scidb package provides an R interface to SciDB.

BugReports https://github.com/Paradigm4/SciDBR/issues
URL http://paradigm4.github.io/SciDBR
Depends R (>= 3.0.0), methods
Imports Matrix, utils, stats, iterators, RCurl, zoo, graphics
License AGPL-3
LazyLoad yes
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-04-16 22:49:07

R topics documented:

  scidb-package ................................................ 3
  aggregate-methods ........................................... 3
  all.equal ..................................................... 6
  antijoin ....................................................... 7
  as.scidb ....................................................... 8
  attribute_rename ............................................. 9
  between ....................................................... 10
  bind .......................................................... 11
  build .......................................................... 12
R topics documented:

c-methods ................................................................. 14
cast ........................................................................ 15
comparison-methods .................................................. 16
cumulate ................................................................. 18
df2scidb ................................................................. 19
diff.scidb ............................................................... 20
dim.scidb ............................................................... 21
dimensions .............................................................. 21
dimension_rename .................................................... 22
dimnames.scidb ........................................................ 23
glm.fit ..................................................................... 24
grand-methods ......................................................... 26
hist-methods .............................................................. 28
hypergeometric-methods .......................................... 29
index_lookup .......................................................... 30
iquery .................................................................. 31
is.scidb ................................................................. 33
is.scidbdf ............................................................. 33
lag ......................................................................... 34
length.scidb ........................................................... 35
merge-methods .......................................................... 35
na.locf-methods ....................................................... 37
names.scidb ............................................................ 38
nrow.scidb ............................................................... 39
Ops.scidb ............................................................... 39
persist ................................................................. 40
predict.glm_scidb .................................................... 41
print.scidb ............................................................. 43
project ................................................................. 43
quantile ................................................................. 44
redimension ............................................................. 45
rename ................................................................. 47
repart ................................................................. 48
reshape-methods ...................................................... 49
schema ................................................................. 50
scidb ................................................................. 51
scidb-class ............................................................ 54
scidbconnect .......................................................... 56
scidbdf-class .......................................................... 57
scidbdisconnect ...................................................... 58
scidbeval ............................................................ 59
scidblist ............................................................... 60
scidbremove ........................................................... 61
scidb_attributes ....................................................... 62
scidb_fisher.test ..................................................... 63
show_commit_log ...................................................... 64
slice .................................................................... 65
sort-methods ........................................................... 66
 aggregate-methods

| str.scidb | ................................. | 67 |
| subarray | .................................. | 67 |
| subset-methods | .................................. | 68 |
| substitute | .................................. | 69 |
| summary.scidb | .................................. | 70 |
| svd | ................................. | 70 |
| sweep-methods | .................................. | 71 |

Index

73

scidb-package

*SciDB interface package.*

Description

The `scidb` package provides basic functions that move data between R and SciDB and issue SciDB queries.

Details

| Package: | scidb |
| Type: | Package |
| Version: | 0.0 |
| Date: | 2012-07-18 |
| License: | GPL3 |
| Depends: | methods |

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Maintainer: B. W. Lewis <blewis@paradigm4.com>

aggregate-methods

*Methods for Function aggregate in Package scidb*

Description

Aggregate a SciDB array object grouped by a subset of its dimensions and/or attributes.
aggregate-methods

Usage

```r
## S4 method for signature 'scidb'
aggregate(x, by, FUN, eval, window, variable_window, unpack)
## S4 method for signature 'scidbdf'
aggregate(x, by, FUN, eval, window, variable_window, unpack)
```

Arguments

- **x**: A scidb or scidbdf object.
- **by**: (Optional) Either a single character string or a list of array dimension and/or attribute names to group by; or a SciDB array reference object to group by. Not required for windowed and grand aggregates—see details.
- **FUN**: A character string representing a SciDB aggregation expression or a reduction function.
- **eval**: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.
- **window**: (Optional) If specified, perform a moving window aggregate along the specified coordinate windows—see details below.
- **variable_window**: (Optional) If specified, perform a moving window aggregate over successive data values along the coordinate dimension axis specified by by—see details below.
- **unpack**: (Optional) If TRUE, return an unpacked SciDB result as a scidbdf dataframe-like object. It’s sometimes useful to set this to FALSE if the aggregated result needs to be joined with another array. Default=FALSE.

Details

Group the scidb, or scidbdf array object x by dimensions and/or attributes in the array. applying the valid SciDB aggregation function FUN expressed as a character string to the groups. Set eval to TRUE to execute the aggregation and return a scidb object; set eval to FALSE to return an unevaluated SciDB array promise, which is essentially a character string describing the query that can be composed with other SciDB package functions.

If an R reduction function is specified for FUN, it will be transliterated to a SciDB aggregate.

The by argument must be a list of dimension names and/or attribute names in the array x to group by, or a SciDB array reference object. If by is not specified and one of the window options is not specified, then a grand aggregate is performed over all values in the array.

The argument by may be a list of dimension names and/or attributes of the array x. Attributes that are not of type int64 will be ‘factorized’ first and replaced by enumerated int64 values that indicate each unique level (this requires SciDB 13.6 or higher).

When by is a SciDB array it must contain one or more common dimensions with x. The two arrays will be joined (using SciDB `cross_join(x, by)` and the resulting array will be grouped by the attributes in the by array. This is similar to the usual R data.frame aggregate method.

Perform moving window aggregates by specifying the optional window or variable_window arguments. Use window to compute the aggregate expression along a moving window specified along
each coordinate axis as window=c(dimension_1_low, dim_1_high, dim_2_low, dim_2_high, ...). Moving window aggregates along coordinates may be applied in multiple dimensions.

Use variable_window to perform moving window aggregates over data values in a single dimension specified by the by argument. See below for examples. Moving window aggregates along data values are restricted to a single array dimension.

Value

A scidbdf reference object.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
# Create a copy of the iris data frame in a 1-d SciDB array named "iris."
# Note that SciDB attribute names will be changed to conform to SciDB
# naming convention.
x <- as.scidb(iris,name=iris)

# Compute averages of each variable grouped by Species
a <- aggregate(x, by="Species", FUN=mean)

# Aggregation by an auxiliary vector (which in this example comes from
# an R data frame)--also note any valid SciDB aggregation expression may
# be used:
y <- as.scidb(data.frame(sample(1:4,150,replace=TRUE)))
a <- aggregate(x, by=y, FUN="avg(Petal_Width) as apw, min(Sepal_Length) as msl")

# Use the window argument to perform moving window aggregates along coordinate
# systems. You need to supply a window across all the array dimensions.
set.seed(1)
A <- as.scidb(matrix(rnorm(20),nrow=5))
# Compute a moving window aggregate only along the rows summing two rows at
# a time (returning result to R). The notation (0,1,0,0) means apply the
# aggregate over the current row (0) and (1) following row, and just over
# the current column (that is, a window size of one).
aggregate(A,FUN="sum(val)" ,window=c(0,1,0,0))
# The above aggregate is equivalent to, for example:
apply(a,2,function(x) x+c(x[-1],0))

# Moving windows using the window= argument run along array coordinates.
# Moving windows using the variable_window= argument run along data values,
# skipping over empty array cells. The next example illustrates the
# difference.

# First, create an array with empty values:
B <- A>0
# Here is what B looks like:
B[]
```
all.equal

Description

Test if two SciDB arrays are equal in content, ignoring SciDB chunk partitioning and overlap differences.

Usage

```r
## S3 method for class 'scidb'
all.equal(target, current, ...)
## S3 method for class 'scidbdf'
all.equal(target, current, ...)
```

Arguments

- `target`: a `scidb` or `scidbdf` object.
- `current`: a `scidb` or `scidbdf` object.
- `...`: optional arguments that might be used in a future version.

Value

Return `TRUE` if the `target` SciDB array has the same dimensions, same attributes and types and same data at the same coordinates as the `current` array. Return `FALSE` otherwise.

Author(s)

A. Poliakov <apoliakov@paradigm4>

See Also

`scidb`
Examples

## Not run:
# Let's build a constant-valued 10x10 array:
a <- build(p1, c(10,10), chunksize=c(5,7))

# The array is equal to itself!
all.equal(a,a)  # (returns TRUE)

# Let's build an identically-shaped and valued array but with different
# array chunk partitioning and overlap:
b <- build(p1, c(10,10), chunksize=c(9,10), overlap=c(1,1))

# all.equal does not care about SciDB chunk sizes or overlap differences:
all.equal(a,b)  # (returns TRUE)

# But differences in value or dimension matter:
all.equal(a, b*2)  # (returns FALSE)

## End(Not run)

Description

Given two SciDB arrays of same dimensionality, return any coordinates that do NOT join.

Usage

antijoin(array1, array2)

Arguments

array1  a scidb array or scidbdf data frame object.
array2  a scidb array or scidbdf data frame object.

Value

NULL is returned when the two arrays join in all coordinates. Otherwise, a scidb array is returned such that for all coordinates, the single attribute shall equal to 1 if those coordinates exist only in array1, or 2 if those coordinates exist only in array2.

Author(s)

A. Poliakov <apoliakov@paradigm4.com>

See Also

scidb
Examples

## Not run:
set.seed(1)

# Generate sparse SciDB arrays
a <- as.scidb(Matrix::sparseMatrix(
    sample(10, 100, replace=TRUE), sample(10, 100, replace=TRUE), x=runif(100))
)
b <- as.scidb(Matrix::sparseMatrix(
    sample(10, 100, replace=TRUE), sample(10, 100, replace=TRUE), x=runif(100))
)

antijoin(a, b)

# Output looks like:
# [1,] 2 . . 1 2 1 . . 2
# [2,] 2 . 2 2 1 1 2 2
# [3,] 2 . 2 . 2 . 1
# [4,] 2 1 . . . 1 . 2
# [5,] . . . . 1 1 . . 2
# [6,] 1 2 . 2 2 . 1 2
# [7,] 1 . 1 . 1 1 1
# [8,] 2 . 1 . 1 1 1 .
# [9,] 2 . 1 . 2 . 2 1.
# [10,] 2 . 2 . 2 2 .

## End(Not run)

---

**as.scidb**

Upload an R matrix or data.frame to a SciDB array.

**Description**

Upload an R matrix or data.frame into SciDB, returning a reference scidb object.

**Usage**

```r
as.scidb(X, name = tmpnam(), chunkSize, overlap, start, gc=TRUE, ...)
```

**Arguments**

- **X**
  A vector, matrix or sparse matrix of double-precision floating point values or a data.frame.

- **name**
  The name of the SciDB array to create.

- **chunkSize**
  A vector of optional SciDB array chunk sizes (a vector of length 1 for data frames and vectors, length 2 for matrices)

- **overlap**
  (Not yet supported)

- **start**
  A vector of SciDB array starting dimension numeric index value or values.

- **gc**
  Set to TRUE to remove SciDB array when R object is garbage collected or R exists. FALSE means SciDB array persists.

- **...**
  additional arguments to pass to df2scidb (see df2scidb).
attribute_rename

Details

Used with a matrix or vector argument, the `as.scidb` function creates a single-attribute SciDB array named `name` and copies the data from `x` into it, returning a `scidb` object reference to the new array. The SciDB array will be 1-D if `x` is a vector, and 2-D if `x` is a matrix.

If `x` is a data.frame, then `as.scidb` creates a one-dimensional multi-attribute SciDB array, with SciDB attributes representing each column of the data.frame. A utility function called `df2scidb` is used for data.frame uploads—the functions `as.scidb` and `df2scidb` are equivalent in this use case.

This function supports double-precision, integer (32-bit), logical, and single-character array attribute types.

Value

A `scidb` reference object.

Note

The `as.scidb` function is not a very efficient way to upload lots of data to SciDB. Consider using the SciDB parallel bulk loader tools for that.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

See Also

`scidb df2scidb`

Examples

```r
## Not run:
X <- matrix(runif(20),5)
A <- as.scidb(X)
as.scidb(iris)
scidblist()
print(A)
## End(Not run)
```

---

attribute_rename | Rename an attribute.

Description

Rename a SciDB array attribute.

Usage

```r
attribute_rename(x, old, `new`, `eval`=FALSE)
```
### Arguments

- **x**: A scidb or scidbdf object.
- **old**: A character string representation of the old attribute name.
- **new**: A character string representation of the new attribute name.
- **eval** (Optional): If true, execute the query and store the result array. Otherwise defer evaluation.

### Value

A new scidb or scidbdf object.

### Note

If you don’t specify the **old** argument, it will default to the attribute names of **x**.

### Author(s)

B. W. Lewis <blewis@paradigm4.com>

### See Also

- `scidb dimension_rename`
- `cast`

### Examples

```r
## Not run:
x <- as.scidb(iris)
y <- attribute_rename(x, "Species", "Group")
print(head(y))
## End(Not run)
```

---

### Description

Use `between` to select contiguous subarrays in indexing operations. The `between` function would not normally be used directly but rather inside bracket indexing operations as shown in the example. This function is designed to support efficient indexing of contiguous subarrays for arrays with non-integer dimensions.

### Usage

`between(a, b)`
The \texttt{bind} function is a wrapper to the SciDB `apply` operator. We chose the name `bind` as this function most closely matches the concept of the R \texttt{cbind} function in data frame contexts.

Similarly to \texttt{cbind}, the \texttt{bind} function adds a new variable (a.k.a. SciDB ‘attribute’) to a SciDB array.

\textbf{Usage}

\begin{verbatim}
bind(X, name, FUN, eval)
\end{verbatim}
Arguments

X A scidb or scidbdf object
name A character name for the new attribute
FUN A SciDB function or expression that defines the values for the new attribute (character)
eval (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Value

A scidbdf object.

Note

Use eval=FALSE to defer expression evaluation for efficiency.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

scidb

Examples

## Not run:
x <- as.scidb(iris)
y <- bind(x, "prod", "Petal.Length * Petal.Width")

# The new scidbdf object y contains a new variable named 'prod':
print(head(y))

## End(Not run)

Description

The build function is a wrapper to the SciDB ‘build’ operator. Operation is similar to the R matrix and array functions. It creates a new single-attribute SciDB array based on the specified parameters.

Usage

build(data, dim, names, type, start, name, chunksize, overlap, gc=TRUE, 'eval')
Arguments

- **data**: A valid SciDB expression (expressed as a character string) or constant to fill the array.
- **dim**: A vector of dimension lengths.
- **names**: An optional vector of attribute and dimension names. Default attribute name is ‘val’ and the dimension names are labeled i,j,... by default.
- **type**: The SciDB type of the array attribute.
- **start**: An optional vector of starting dimension coordinate indices. Must match the length of the dim vector.
- **name**: Optional name of the SciDB array. An automatically generated name is used by default.
- **chunksize**: An optional vector of dimension chunk sizes. Must match the length of the dim vector.
- **overlap**: An optional vector of dimension overlap values. Must match the length of the dim vector.
- **gc**: gc=TRUE (the default) removes the array when corresponding R objects are garbage collected. Set gc=FALSE to preserve arrays.
- **eval**: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Value

A scidb object.

Note

Use eval=FALSE to defer expression evaluation for efficiency.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

scidb

Examples

```r
## Not run:
y <- build(pi, c(5,3))
print(head(y))
```

## End(Not run)
Description

Concatenate SciDB arrays along one coordinate axis.

Usage

```r
## S4 method for signature 'scidbdf'
c(x,y,'eval'=FALSE)
```

Arguments

- `x`: A scidbdf object.
- `y`: A scidbdf object with the same attribute pattern as `x`.
- `eval`: (Optional) If true, execute the query. Otherwise defer evaluation.

Details

Concatenate the SciDB arrays along their single dimension. The output array will include the entries of array `y` below the entries of `x`. The input arrays must have the same number of attributes and attribute types.

Value

A scidbdf reference object.

Note

This method will soon be extended to n-d arrays and to accept a list of more than two input arrays.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
data("iris")
x <- as.scidb(iris)
c(x,x)

## End(Not run)
```
**cast**

*Low-level SciDB cast operator*

**Description**

Apply the low-level SciDB cast operator to a SciDB array.

**Usage**

`cast(x, schema, `eval`=FALSE)`

**Arguments**

- **x**
  
  A scidb or scidbdf object.

- **schema**
  
  A character string representation of a SciDB array schema.

- **eval**
  
  (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

**Value**

A new scidb or scidbdf object.

**Note**

The low-level SciDB cast operator can be used to change attribute and dimension names of an array.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

**See Also**

- `scidb attribute_rename dimension_rename`

**Examples**

```r
## Not run:
x <- as.scidb(iris)
y <- cast(x,"<a:double,b:double,c:double,e:double,f:string>[row=1:150,150,0]")

print(head(y))

## End(Not run)
```
**Comparison Methods**

**Description**

The binary operators described here perform comparison operations that return a sparse SciDB array of the same shape as the input array but only containing entries where the comparison evaluates to TRUE.

Compare this with standard R comparison operators on SciDB arrays, that return an array populated with TRUE or FALSE values.

Masked comparison arrays can be used to efficiently index other SciDB arrays.

**Usage**

```r
## S4 method for signature 'scidb'
x %<% y
## S4 method for signature 'scidbdf'
x %<% y
## S4 method for signature 'scidb'
x %>% y
## S4 method for signature 'scidbdf'
x %>% y
## S4 method for signature 'scidb'
x %<=% y
## S4 method for signature 'scidbdf'
x %<=% y
## S4 method for signature 'scidb'
x %>=% y
## S4 method for signature 'scidbdf'
x %>=% y
## S4 method for signature 'scidb'
x %==% y
## S4 method for signature 'scidbdf'
x %==% y
```

**Arguments**

- `x` A scidb or scidbdf object.
- `y` A scalar value.

**Details**

The comparisons outlined here are limited to scalars. A future version will include element-wise comparison between arrays. For now, use the `bind` and `merge` functions to manually perform element-wise comparisons.
The standard R comparison operators by convention return an array of the same size as \( x \) with TRUE or FALSE values indicating the result of the comparison for each cell. That kind of output is especially useful in subsequent aggregations, for example.

The alternate comparison methods outlined here return an array of the same shape as \( x \), but masked to only contain values in cells where the condition evaluates to TRUE. Remaining cells are empty. This kind of comparison method is useful to quickly extract the values that meet the condition, and also to use the masked array as an index to subset other SciDB arrays.

The examples below illustrate each kind of comparison operator.

Value

A scidb or scidbdf array of the same shape as \( x \).

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
# Not run:
> set.seed(1)
> x=as.scidb(rnorm(10))
> x[]
# [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078 -0.8204684
# [7]  0.4874291  0.7383247  0.5757814 -0.3053884
> (x < 0)[]
# [1] TRUE FALSE TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE
> Hx as_sparse_vector HnnzOlength O of class "dsparseVector"
# [1] -0.6264538 -0.8356286 . . . . . . .
# [7] . . . . . . . . .

> Filter("val < 0", x)[[]]
#sparse vector (nnz/length = 4/10) of class "dsparseVector"
# [1] -0.6264538 -0.8356286 . . . . . .
# [7] . . . . . . . .

# Sparse filtered output is useful to use to index SciDB arrays. The next
# example selects just the entries of the array that meet the condition:
> x[x]
# [1] -0.6264538 -0.8356286 -0.8204684 -0.3053884

# The TRUE/FALSE output array is useful to aggregate by groups defined
# by the condition. The next example computes the mean of the entries
# that are less than zero, and the mean of the entries that are greater
# than or equal to zero:
> aggregate(x, by=(x<0), FUN=mean)[]
# condition_index val_avg condition
#0 0 0.6516612 false
#1 1 -0.6469848 true
```
Description

Use `cumulate` function to compute running operations along data, for example cumulative sums.

Usage

```
cumulate(x, expression, dimension, eval)
```

Arguments

- **x**: A scidb or scidbdf object
- **expression**: Any valid SciDB aggregate expression, expressed as a character string
- **dimension**: An optional array dimension name (character) to run along. The default is to use the first dimension.
- **eval**: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Value

A scidb object.

Note

Use `eval=FALSE` to defer expression evaluation for efficiency.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

`aggregate`

Examples

```
## Not run:
x <- as.scidb(iris)
y <- cumulate(x, "sum(Petal_Width)")
print(head(y))
## End(Not run)
```
Copy a data frame into a SciDB array.

Description

Copy a data frame into a new 1-D SciDB array.

Usage

```r
df2scidb(x, name = tmpnam(), dimlabel = "row", chunkSize,
            rowOverlap = 0L, types = NULL, nullable, schema_only = FALSE, gc, start)
```

Arguments

- `x`: A data frame.
- `name`: The SciDB array name, defaults to an automatically-generated name.
- `dimlabel`: Name the SciDB dimension.
- `chunkSize`: The SciDB chunk size.
- `types`: An optional vector explicitly specifying the SciDB attribute types. Length must match the number of columns of the data frame.
- `nullable`: An optional vector indicating the SciDB nullable property of each attribute. Length must match the number of columns of the data frame.
- `schema_only`: If TRUE, return a string that would represent the SciDB array schema without uploading data to SciDB.
- `gc`: Optional logical value. If TRUE, then resulting SciDB array will be garbage-collected when the R variable referencing it is. The default value is FALSE.
- `start`: Optional integer starting index value. Defaults to 1.

Details

`df2scidb` is a workhorse utility function that transfers an R data frame into a 1-D SciDB array via intermediate CSV formatting. The columns of the data frame correspond to attributes in the SciDB array. The `iquery` function returns query results using a similar method as R data frames.

Value

NULL is invisibly returned. SciDB errors are propagated as R error conditions.

Author(s)

B. W. Lewis <blewis@paradigm4.com>
Examples

```r
## Not run:
df2scidb(iris)
scidblist()
head(iquery("scan(iris)", return=TRUE))

## End(Not run)
```

Description

Apply a finite difference operator to a numeric series.

Usage

```r
## S3 method for class 'scidb'
diff(x, lag=1, ...)
```

Arguments

- **x**: A scidb object with a numeric attribute.
- **lag**: An integer lag value (see description).
- **...**: Optional arguments, not used.

Value

If `x` is a vector of length `n` and starting index 1, then the computed result is equal to the difference 
`x[(1+lag):n] - x[1:(n-lag)]`.

If `x` is a matrix then the difference operations are carried out on each column separately.

SciDB missing (NULL) and NaN values propagate.

Author(s)

B. W. Lewis <blewis@paradigm4.com>
dim.scidb

---

**Description**

dim

**Usage**

```r
## S3 method for class 'scidb'
dim(x)
```

**Arguments**

- `x` A `scidb` object.

**Value**

Vector of matrix dimensions or NULL.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

---

dimensions

---

**Description**

Return a vector of SciDB array dimension names.

**Usage**

`dimensions(x)`

**Arguments**

- `x` An object of class `scidb` or `scidbdf`

**Value**

A character vector of SciDB array dimension names.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>
See Also

scidb

Examples

```r
## Not run:
# Upload data to create a 10x10 SciDB matrix
A <- as.scidb(matrix(rnorm(100),10))
dimensions(A)

## End(Not run)
```

dimension_rename  Rename a dimension.

Description

Rename a SciDB array dimension.

Usage

dimension_rename(x, old, `new`, `eval`=FALSE)

Arguments

- **x**  A scidb or scidbdf object.
- **old**  A character string representation of the old dimension name.
- **new**  A character string representation of the new dimension name.
- **eval**  (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Value

A new scidb or scidbdf object.

Note

If you don’t specify the old argument, it defaults to the dimension names of x.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

See Also

scidb attribute_rename cast
Examples

```r
## Not run:
x <- as.scidb(iris)
y <- dimension_rename(x, "row", "i")

str(y)
## End(Not run)
```

### dimnames.scidb

#### Description

Return a list of dimension label arrays.

#### Usage

```r
## S3 method for class 'scidb'
dimnames(x)
```

#### Arguments

- `x` SciDB array reference object.

#### Details

The scidb package can label SciDB array coordinate systems with indices from other SciDB arrays.

#### Value

A list of dimension label arrays.

#### Author(s)

B. W. Lewis <blewis@paradigm4.com>

#### See Also

- `dimnames<-`, `rownames`, `colnames`

#### Examples

```r
## Not run:
A <- as.scidb(matrix(rnorm(20), nrow=5))
colnames(A) <- as.scidb(data.frame(letters[1:4]))
dimnames(A)

## End(Not run)
```
glm.fit  

Fitting generalized linear models

Description

glm.fit is used to fit generalized linear models specified by a model matrix and response vector. 
glm_scidb is a simplified interface for scidbdf objects similar (but much simpler than) glm.

Usage

## S4 method for signature 'scidb'
glm.fit(x, y, weights=NULL, family=gaussian())
glm_scidb(formula, data, family=gaussian(), weights=NULL)
model_scidb(formula, data, factors=NULL)

Arguments

x  
a model matrix of dimension 'n * p'.
y  
a response vector of length 'n'.
formula  
an object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted. See details for limitations.
data  
an object of class scidbdf.
weights  
an optional vector of 'prior weights' to be used in the fitting process. Should be 'NULL' or a numeric or scidb vector.
family  
a description of the error distribution and link function to be used in the model, supplied as the result of a call to a family function.
factors  
a list of factor encodings to use in the model matrix. See details.

Details

The glm_scidb function works similarly to a limited version of the usual glm function, but with a scidbdataframe-like SciDB array instead of a standard data.frame.

Formulas in the glm_scidb function may only refer to variables present in the data scidbdf object. And the indicated response must refer to a single-column response term in the data (the two-column response form is not accepted). Formulas may only list variables explicitly defined in the data. That means that you should bind interaction and transformed terms to your data before invoking the function.

Categorical (factor) variables in the data must be represented as strings. They will be encoded as treatment-style contrast variables with the first listed value set to the baseline value. No other automated contrast encodings are available yet (you are free to build your own model matrix and use glm.fit for that). All other variables will be coerced to double-precision values.

Use the model_scidb function to build a model matrix from a formula and a scidbdf data frame-like SciDB array. The matrix is returned within an output list as a sparse SciDB matrix of class scidb with character string variables encoded as treatment contrasts as described above. If you already have a list of factor-level codes for categorical variables (for example from the output of glm_scidb, you can supply that in the factor argument. See help for predict for an example.
The `glm.fit` and `glm_scidb` functions return a list of model output values described below. The `glm_scidb` method uses an S3 class to additionally overload nice printing and summary methods.

1. `coefficients` model coefficient vector (SciDB array)
2. `stderr` vector of model coefficient standard errors (SciDB array)
3. `tval` vector of model coefficient t ratio values using estimated dispersion value (SciDB array)
4. `pval` vector of two-tailed p-values corresponding to the t ratio based on a Student t distribution. (It is possible that the dispersion is not known and there are no residual degrees of freedom from which to estimate it. In that case the estimate is 'NaN'.)
5. `aic` a version of Akaike’s An Information Criterion value.
6. `null.deviance` the deviance for the null model, comparable with deviance.
7. `res.deviance` up to a constant, minus twice the maximized log-likelihood.
8. `dispersion` For binomial and Poison families the dispersion is fixed at one and the number of parameters is the number of coefficients. For gaussian, Gamma and inverse gaussian families the dispersion is estimated from the residual deviance, and the number of parameters is the number of coefficients plus one. For a gaussian family the MLE of the dispersion is used so this is a valid value of AIC, but for Gamma and inverse gaussian families it is not. Other families set this value to `NA`.
9. `df.null` the residual degrees of freedom for the null model.
10. `df.residual` the residual degrees of freedom.
11. `converged` FALSE if the model did not converge.
12. `totalObs` total number of observations in the model.
13. `nOK` total number of observations corresponding to nonzero weights.
14. `loglik` converged model log-likelihood value.
15. `rss` residual sum of squares.
16. `iter` number of model iterations.
17. `weights` vector of weights used in the model (SciDB array).
18. `family` model family function.
19. `y` response vector (SciDB array).
20. `x` model matrix (SciDB array).
21. `factors` a list of factor variable levels (SciDB arrays) or NULL if no factors are present in the data.

`model_scidb` returns an output list with:

- `formula` the model forumual.
- `model` the model matrix (SciDB array).
- `response` the model response vector (SciDB array).
- `names` an R character vector of variable names in the model matrix.
- `intercept` a logical value; if TRUE the model includes an intercept term.
- `factors` a list of factor variable levels (SciDB arrays) or NULL if no factors are present in the data.
**Author(s)**

B. W. Lewis<blewis@paradigm4.com>

**See Also**

`scidb predict glm_scidb`

**Examples**

```r
## Not run:
# Using glm.fit
x <- as.scidb(matrix(rnorm(5000*20),nrow=5000))
y <- as.scidb(rnorm(5000))
M <- glm.fit(x, y)
coef(M)[]

# Using glm_scidb (similar to glm)
# From the 'glm' help:
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
d.AD <- data.frame(treatment, outcome, counts)
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson(),data=d.AD)
summary(glm.D93)

# Compare with:
d.AD_sci = as.scidb(d.AD)
glm.D93_sci = glm_scidb(counts ~ outcome + treatment, family = poisson(), data=d.AD_sci)
summary(glm.D93_sci)

## End(Not run)
```

---

**grand-methods**  
**Miscellaneous methods from package scidb**

**Description**

The methods described here compute a global aggregate on a SciDB array object and return a scalar value to R.

**Usage**

```r
## S4 method for signature 'scidb'
mean(x)
## S4 method for signature 'scidb'
mean(x)
## S4 method for signature 'scidb'
median(x)
```
Arguments

x | A scidb or scidbdf object.

Details

sum computes the global sum of the SciDB array elements (applies only to numeric types); mean computes the arithmetic average of the SciDB array elements; median computes the median of the SciDB array elements; min computes the minimum of the SciDB array elements; max computes the maximum of the SciDB array elements; count returns the number of non-empty cell values in the SciDB array; sd computes the standard deviation of the SciDB array elements (numeric only); var computes the variance of the SciDB array elements (numeric only).

Value

The methods documented here always return a scalar value.

Note

SciDB convention dictates that global aggregates only apply to the first listed attribute of a multi-attribute SciDB array (e.g., a scidbdf data frame-like object). Use `project(array, attribute)` to select a different attribute—see the examples.
Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
data("iris")
x <- as.scidb(iris)
sum(x$Petal.Length)

## End(Not run)
```

Description

The generic function `hist` computes a histogram of the given data values. If `plot = TRUE`, the resulting object of class `histogram` is plotted by `plot.histogram`, before it is returned.

Usage

```r
## S4 method for signature 'scidb'
hist(x, breaks=10, right=FALSE, materialize=TRUE, `eval`=FALSE, `plot`=TRUE, ...)
## S4 method for signature 'scidbdf'
hist(x, breaks=10, right=FALSE, materialize=TRUE, `eval`=FALSE, `plot`=TRUE, ...)
```

Arguments

- `x`: A scidb or scidbdf object.
- `breaks`: a single number giving the number of cells for the histogram.
- `right`: logical; if `TRUE`, the histogram cells are right-closed (left open) intervals.
- `materialize`: logical; if `TRUE`, the histogram results are materialized into R vectors and returned as a standard object of class `histogram`. Otherwise the results are returned as SciDB objects.
- `eval`: Optional, logical; If TRUE, execute the query and store the result array. Otherwise defer evaluation.
- `plot`: Optional, logical; If TRUE, plot result before returning.
- `...`: Optional; additional arguments for plotting.

Details

This histogram function only supports equidistant breaks. If `right = TRUE` (default), the histogram cells are intervals of the form `(a, b]`, i.e., they include their right-hand endpoint, but not their left one.
Value

A histogram object, which is a list of components:

- breaks: the n+1 cell boundaries.
- counts: n integers; for each cell, the number of x[] inside.
- density: estimated density values.
- mids: the n cell midpoints.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Description

Distribution and quantile functions for the hypergeometric distribution.

Usage

```r
## S4 method for signature 'scidb_or_scidbdf'
phyper(x, q, m, n, k, new="p","eval"=FALSE)
## S4 method for signature 'scidb_or_scidbdf'
quhyper(x, p, m, n, k, new="q","eval"=FALSE)
```

Arguments

- `x`: A scidb or scidbdf object.
- `q`: A SciDB attribute name (character) or numeric constant representing the quantile, for example the number of white balls drawn without replacement from an urn which contains both black and white balls.
- `m`: A scidb attribute name or numeric constant representing the number of white balls in the urn.
- `n`: A scidb attribute name or numeric constant representing the number of black balls in the urn.
- `k`: A scidb attribute name or numeric constant representing the number of balls drawn from the urn.
- `p`: A scidb attribute name or numeric constant representing probability between 0 and 1.
- `new`: A character name for the result attribute.
- `eval`: (Optional) If true, execute the query. Otherwise defer evaluation.
Details

The hypergeometric distribution is used for sampling without replacement. The density of this distribution with parameters \( m, n \) and \( k \) is given by

\[
p(x) = \binom{m}{x} \binom{n}{k-x} / \binom{m+n}{k}
\]

for \( x = 0, \ldots, k \).

The quantile is defined as the smallest value \( x \) such that \( F(x) \geq p \), where \( F \) is the distribution function.

\( \text{phyper} \) gives the distribution function, \( \text{qhyper} \) gives the quantile function.

Value

A scidb, scidbdf reference object.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
# Write me

## End(Not run)
```

```
index_lookup
```

Description

The \texttt{index\_lookup} function is a wrapper to the SciDB ‘index\_lookup’ operator. It produces a new SciDB array that joins the unique indices defined in the array \( I \) with values looked up in array \( X \) for attribute \( \texttt{attr} \). Use the \texttt{index\_lookup} with the unique and sort functions.

Usage

\[
\text{index\_lookup}(X, I, \texttt{attr}, \texttt{new\_attr}, \texttt{eval}=\text{FALSE})
\]

Arguments

\begin{itemize}
\item \texttt{X} A scidb or scidbdf object
\item \texttt{I} A scidb or scidbdf object
\item \texttt{attr} A character string attribute name from the \( X \) array
\item \texttt{new\_attr} An optional character name for the new attribute, defaults to \( \texttt{attr} + \_index \)
\item \texttt{eval} (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.
\end{itemize}
Value
A scidb or scidbdf object.

Note
Use eval=FALSE to defer expression evaluation for efficiency.

Author(s)
B. W. Lewis<blewis@paradigm4.com>

See Also
scidb

Examples
```r
## Not run:
data("iris")
x <- as.scidb(iris)

# Create a unique list of elements of the "Species" attribute.
# Note that we choose to defer evaluation of this expression.
y <- unique(sort(project(x,"Species")), eval=FALSE)

# Append a new attribute to the array x called "Species_index" that
# enumerates the unique values of the "Species" attribute:
z <- index_lookup(x, y, "Species", eval=FALSE)

print(head(z))

## End(Not run)
```

iquery
---

**Simple SciDB query tool**

Description
Issue SciDB queries and optionally return output in a data frame.

Usage
```r
iquery(query, `return` = FALSE, afl = TRUE, iterative = FALSE, n = 10000, excludecol, ...)
```
Arguments

- **query**: A SciDB query string (character). Separate multiple queries with semicolons.
- **return**: Set to TRUE to return output. Otherwise don’t return query output. Only available when `afl`=TRUE.
- **afl**: TRUE indicates query is in AFL form, FALSE indicates AQL.
- **iterative**: Set to TRUE to return a result iterator. FALSE returns entire result at once.
- **n**: Maximum number of rows to return when iterating through results. Set to Inf to return everything.
- **excludecol**: An optional numeric range of columns to exclude from iterative results (only applies when `iterative`=TRUE).
- **...**: Options passed on to `read.table` used to parse results.

Details

The `iquery` function is a simple analog of the command-line SciDB `iquery` program.

Value

If `return`=TRUE, return the query result in data frame form (similar to the command-line `-olcsv+` output option).

If `return`=FALSE, return the query ID number.

SciDB errors encountered during query processing are propagated to R and can be handled with normal R error handling mechanisms.

Set `iterative`=TRUE to return a result iterator. Use the iterator `nextElem` function to iteratively return results, a maximum of n results at a time. See help in the iterators package for examples and options.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
iquery("list('instances')", return=TRUE)

# A simple example that iterates through results using foreach
# Build an array with 1 million numbers from zero to 1.
iquery("store(build(<x:double>[i=1:1000000,100000,0],i/1000000),X")
# Apply a function and return result in an iterator:
i <- iquery("apply(X, y, sin(x))", return=TRUE, iterative=TRUE)

# Sum up x and y (and dimension i too)
library("foreach")
foreach(j=i, .combine=function(...){colSums(rbind(...))})

# Compare with the much faster equivalent inside SciDB:
```
is.scidb

```r
iquery("aggregate(apply(X, y, sin(x)),sum(x),sum(y))", return=TRUE)
```

## End (Not run)

---

### Description
Is this a SciDB array object?

### Usage

```r
is.scidb(x)
```

### Arguments

- `x` Something.

### Value
Returns TRUE if `x` is a SciDB object.

### Author(s)
B. W. Lewis<blewis@paradigm4.com>

---

### Description
Is this a SciDB array data.frame-like object?

### Usage

```r
is.scidbdf(x)
```

### Arguments

- `x` Something.

### Value
Returns TRUE if `x` is a SciDB object.

### Author(s)
B. W. Lewis<blewis@paradigm4.com>
**Description**

Lag or lead a time series

**Usage**

```r
## S4 method for signature 'scidb'
lag(x, k=1, dim=1, `eval`=FALSE)
```

**Arguments**

- `x`: A scidb array object.
- `k`: Coordinate system units to lag by.
- `dim`: SciDB array dimension to lag along.
- `eval`: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

**Value**

A scidb array object with identical schema as `x`, but whose values are shifted along the coordinate axis `dim` by the lag value `k`.

**Author(s)**

B. W. Lewis<blewis@paradigm4.com>

**See Also**

scidb

**Examples**

```r
## Not run:
x <- build("i", dim=10, names=c("x","i"), start=1, type="double")
y <- lag(x)
merge(x,y,by="i",all=TRUE)
## End(Not run)
**length.scidb**

---

**Description**

Return the total number of array elements of a SciDB array object.

**Usage**

```r
## S3 method for class 'scidb'
length(x)
```

**Arguments**

- `x` A scidb object.

**Value**

Numeric length of the object in elements.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

---

**merge-methods**

*Methods for Function merge in Package scidb*

---

**Description**

SciDB merge, cross_join, and join operations.

**Usage**

```r
## S4 method for signature 'scidb'
merge(x,y, by=intersect(dimensions(x),dimensions(y)), by.x, by.y, merge, eval)
## S4 method for signature 'scidbdf'
merge(x,y, by=intersect(dimensions(x),dimensions(y)), by.x, by.y, merge, eval)
```
Arguments

- **x**: A scidb or scidbdf reference object.
- **y**: A scidb or scidbdf reference object.
- **by**: (Optional) Vector of common dimension or attribute names to join on. See details below.
- **by.x**: (Optional) Vector of dimension or attribute names of array x to join on. See details.
- **by.y**: (Optional) Vector of dimension or attribute names of array y to join on. See details.
- **merge**: (Optional) If true, perform a SciDB merge operation instead of join.
- **eval**: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Details

Only one of either by or both by.x and by.y may be specified. If none of the by.x,by.y arguments are specified, and by=NULL the result is the Cartesian product of x and y. The default value of by performs a cross_join or join along common array dimensions.

If only by is specified, the dimension names or attribute name in by are assumed to be common across x and y. Otherwise dimension names or attribute names are matched across the names listed in by.x and by.y, respectively.

If dimension names are specified and by contains all the dimensions in each array, then the SciDB join operator is used, otherwise SciDB’s cross_join operator is used. In each either case, the output is a cross product set of the two arrays along the specified dimensions.

If by or each of by.x and by.y list a single dimension name, the indicated attributes will be lexicographically ordered as categorical variables and SciDB will redimension each array along new coordinate systems defined by the attributes, and then those redimensioned arrays will be joined. This method limits joins along attributes to a single attribute from each array. The output array will contain additional columns showing the attribute factor levels used to join the arrays.

This method is limited to SQL-like ‘natural joins’, a special case of inner joins corresponding to the all=FALSE case in the standard R merge function. A future version of this package will include additional join cases.

Specify merge=TRUE to perform a SciDB merge operation instead of a SciDB join.

The various SciDB join operators generally require that the arrays have identical partitioning (coordinate system bounds, chunk size, etc.) in the common dimensions. The merge method attempts to rectify SciDB arrays along the specified dimensions as required before joining.

Value

A scidb or scidbdf reference object.

Author(s)

B. W. Lewis <blewis@paradigm4.com>
Examples

```r
## Not run:
# Create a copy of the iris data frame in a 1-d SciDB array named "iris."
# Note that SciDB attribute names will be changed to conform to SciDB
# naming convention.
# x <- as.scidb(iris, name="iris")

a <- x$Species
b <- x$Petal_Length

c <- merge(a, b, by="row")
merge(b, b, by="row", merge=TRUE)

## End(Not run)
```

Description

Generic function for replacing each missing or empty value with the most recent non-missing value prior to it.

Usage

```r
## S4 method for signature 'scidb'
na.locf(object, along=dimensions(object)[1], eval=FALSE)
```

Arguments

- **object**: A scidb or scidbdf object.
- **along**: A single SciDB array dimension name to replace missing values along.
- **eval** (Optional): If true, execute the query and store the result array. Otherwise defer evaluation.

Details

Unlike the usual `na.locf` function from the zoo package, the SciDB `na.locf` function fills in both missing (SciDB NULL values) and empty (SciDB sparse) values with the last non-missing and non-sparse value along the indicated dimension.

Time series represented in SciDB are often sparse arrays. The `na.locf` function defines a convenient way to fill in all missing values along the time coordinate axis down to the time resolution.

Caution! The output array is a mostly dense, filled-in version of the input array. If the time resolution is very fine and the input array very sparse, then the output array can be huge. Consider using `regrid` first on very fine time scales to reduce their resolution.
Value

A scidb reference object.

Note

The default na.locf method in the zoo package unfortunately overrides this function (it uses ANY in its method signature). If you need to use SciDB arrays and the zoo package, prefix SciDB’s version with scidb::na.locf(...).

Author(s)

B. W. Lewis <blewis@paradigm4.com>

---

### names.scidb

#### Dimension names

**Description**

Return 2nd dimension names of a SciDB array with two or more dimensions, or just the single dimension names of a 1-d SciDB array reference object. Warning! Huge SciDB arrays can return too much. See scidb for an example alternative approach.

**Usage**

```r
## S3 method for class 'scidb'
names(x)
```

**Arguments**

- `x` A SciDB array reference object.

**Value**

Returns a vector of dimension names.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>
**nrow.scidb**

*Number of rows or columns of a SciDB matrix.*

**Description**

Return the number of rows of a SciDB matrix (2-D array).

**Usage**

```r
## S3 method for class 'scidb'
nrow(x)
## S3 method for class 'scidb'
ncol(x)
```

**Arguments**

- `x` A scidb object.

**Value**

The number of rows of the matrix.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

---

**Ops.scidb**

*Binary scidb object operations.*

**Description**

Binary scidb object operations.

**Usage**

```r
## S3 method for class 'scidb'
Ops(e1, e2)
```

**Arguments**

- `e1` A scalar, vector, or matrix value, or SciDB array reference.
- `e2` A scalar, vector, or matrix value, or SciDB array reference.

**Value**

A scidb or scidbdf object.
Persist

Description

Mark a scidb or scidbdf object and any SciDB arrays in its dependency graph persistent.

Usage

persist(x, remove = FALSE, ...)

Arguments

x A scidb or scidbdf object
remove A logical value. See details.
... Optional arguments not used by the default method, but reserved for future use.

Details

Most scidb objects are ephemeral unless the gc=FALSE option is explicitly selected. Use the persist function with remove=FALSE to mark an array and all of its dependencies persistent. This is useful when you want to save a scidb object to an R data file and make sure its data is there when re-loaded.

Specify remove=TRUE to 'unpersist' an object and all of its dependencies.

Value

NULL is invisibly returned. Use this function for its side effects.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

scidb
predict.glm_scidb

Examples

```r
# Not run:
x <- as.scidb(iris)
y <- unique(x$Species)
persist(y)
save(y, file="y.rdata")

# Now, *both* x and y can be removed without deleting the backing SciDB
# array. Without persist or setting gc=FALSE on x, the array would have
# been removed.
rm(x, y)
gc()

# Since the array is still there, we can get it back.
load("y.rdata")
y[]

# End(Not run)
```

---

**predict.glm_scidb**

**Prediction for SciDB GLM fits**

---

**Description**

Obtains predictions and optionally estimates standard errors of those predictions from a fitted generalized linear model object.

**Usage**

```r
## S3 method for class 'glm_scidb'
predict(object, ...)  
```

**Arguments**

- `object` a `glm_scidb` model object.
- `...` optional arguments `newdata`, `type`, and `se.fit` (see details).

**Details**

Optional arguments:

- `newdata` a scidbdf SciDB data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
- `type` the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and `type = "response"` gives the predicted probabilities.
- `se.fit` logical switch indicating if standard errors are required.
Value

If `se.fit = FALSE`, a vector of predictions.
If `se.fit = TRUE`, a list with components:

- **fit**: Predictions, as for `se.fit = FALSE`.
- **se.fit**: Estimated standard errors.
- **residual.scale**: A scalar giving the square root of the dispersion used in computing the standard errors.

Author(s)

B. W. Lewis <blewis@paradigm4.com>, adapted from the R `predict.glm` documentation.

See Also

`scidb glm_scidb`

Examples

```r
## Not run:
## example adapted from Venables and Ripley (2002, pp. 190-2.):

# In R:
ldose <- rep(0:5, 2)
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor(rep(c("M", "F"), c(6, 6)))
data <- data.frame(sex, ldose)
data <= Reduce(rbind,
    lapply(1:length(numdead),
        function(j) rbind(cbind(alive=1, data[j][rep(1, numdead[j])],
            cbind(alive=0, data[j][rep(1, 20-numdead[j])])))
)
rownames(data) <= NULL

r_model <= glm(alive ~ sex + ldose - 1, family=binomial(), data=data)

# Now in SciDB:
data_scidb <= as_scidb(data)
str(data_scidb)
scidb_model <= glm_scidb(alive ~ sex + ldose - 1, family=binomial(), data=data_scidb)

# New data for prediction:
ld <= seq(0.5, 0.1)
newdata <= as_scidb(data.frame(ldose=ld, sex=rep("M", length(ld)))))
head(newdata)
pred_scidb = predict(scidb_model, newdata=newdata, type="response")
head(pred_scidb)

require("graphics")
plot(c(1,32), c(0,1), type = "n", xlab = "dose",
    ylab = "prob", log = "x")
```
print.scidb

Description
Print a summary of a SciDB array object.

Usage
## S3 method for class 'scidb'
print(x, ...)

Arguments
x A scidb array object.
... further arguments passed to or from other methods.

Value
Printed summary output.

Author(s)
B. W. Lewis <blewis@paradim4.com>

project

Description
The project function is a wrapper to the SciDB ‘project’ operator. It creates a new array that is a subset of the input array limited to a set of specified attributes.

Usage
project(X, attributes, eval)
quantile

Arguments

\( X \)  
A scidb or scidbdf object

attributes  
A character vector of attributes to project on to

eval  
(Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Value

A scidb object.

Note

Use eval=FALSE to defer expression evaluation for efficiency.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

scidb

Examples

```r
## Not run:
x <- as.scidb(iris)
y <- project(x, c("Sepal_Length","Petal_Length","Species"))
print(head(y))
## End(Not run)
```

---

quantile  
Sample Quantiles

Description

Compute sample quantiles of a single-attribute SciDB array.

Usage

```r
## S3 method for class 'scidb'
quantile(x, probs=seq(0,1,0.25), type=7, ...)
## S3 method for class 'scidbdf'
quantile(x, probs=seq(0,1,0.25), type=7, ...)
```
redimension

Arguments

- **x**: a scidb or scidbdf object.
- **probs**: a numeric vector of probabilities with value in [0,1].
- **type**: either 1 or 7, selecting the two available quantile algorithms (see help for `quantile` for more details).
- **...**: optional arguments that might be used in a future version.

Details

If the input array contains more than one attribute, the first numeric attribute will be chosen (with a warning). Use `project` to select a single attribute.

The SciDB quantile function only supports types 1 and 7 at the time of this writing. Quantile type 1 can be used with non-numeric values, as long as they can be ordered. See `quantile` for much more detail on the calculation.

Value

A two-attribute SciDB data frame-like array with `length(probs)` rows is returned. The first attribute contains `probs` and the second attribute the sample quantiles.

Author(s)

A. Poliakov <apoliakov@paradigm4>

See Also

- `quantile`

Examples

```r
## Not run:
x <- as.scidb(rnorm(1000))
quantile(x)[]

quantile(x,type=1)[]

## End(Not run)
```
Usage
redimension(x, schema, dim, FUN, `eval`=FALSE)

Arguments
x A SciDB array object of class scidb or scidbdf.
schema An optional SciDB array object of class scidb, scidbdf, or a character string representation of the output array schema.
dim An optional character vector or list of new dimension names from the union of dimension and attribute names of x. Exactly one of the s and dim arguments must be specified.
FUN An optional reduction function applied when redimension collapses multiple values.
eval (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Details
Redimension is a core SciDB operation. It can change the dimensionality, shape, and partitioning of arrays, and transform array attributes into array dimensions and vice versa. Redimension can also apply reduction functions to values when dimensions are removed, similarly to grouped aggregates.
The R package redimension function presents several forms. The most direct form takes a SciDB array reference x and an desired output schema s and directly applies the SciDB redimension operator.
Alternatively, users may specify a character vector or list of dim values that represent new array coordinate axes. These values should be a subset of the union of attributes and dimension names in the input array x. Note that they must also be valid int64 types.
When redimension reduces the dimensionality of an array, it’s likely that multiple values may fall into the same output array cell. When this occurs, SciDB’s default behavior randomly selects one of the possible values for output. Alternatively, users may specify a reduction function in the FUN argument or explicitly specify reductions using SciDB syntax in the schema argument. Note that the indicated reduction function will be applied to all the attributes. If you need more control over the reduction, explicitly specify the output schema instead.

Value
A scidb object.

Note
Use eval=FALSE to defer expression evaluation for efficiency. Results can be composed to form complex query expressions that can be evaluated in one transaction.

Author(s)
B. W. Lewis<blewis@paradigm4.com>
rename

See Also

scidb

Examples

```r
## Not run:
# Upload iris to SciDB:
x <- as.scidb(iris)

# bind an example new 'class' column:
y <- bind(x, "class", "iif(Petal_Width>2, int64(1), 0)")
zz <- redimension(y, dim="class", FUN=max)

## End(Not run)
```

rename

Rename a SciDB array.

Description

Rename a SciDB array, returning a new reference object and optionally setting persistence of the object.

Usage

```r
rename(A, name=A@name, gc)
```

Arguments

- `A`: A scidb or scidbdf object.
- `name`: An optional new name for the object.
- `gc`: An optional logical value indicating object persistence. If TRUE, then the backing SciDB array will be deleted by the R garbage collector when corresponding R references are deleted.

Value

A new scidb or scidbdf object.

Author(s)

B. W. Lewis <blewis@paradigm4.com>
Description

The `repart` function is a wrapper to the SciDB `repart` operator.

Usage

`repart(x, schema, upper, chunk, overlap, eval=FALSE)`

Arguments

- `x` A SciDB array object of class scidb or scidbf.
- `schema` An optional character array schema.
- `upper` An optional vector of new array upper bounds.
- `chunk` An optional vector of new array chunk sizes.
- `overlap` An optional vector of new array overlap values.
- `eval` (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Value

A scidb object.

Note

Repartition SciDB array data by changing the array chunk sizes and/or overlap values. This function is also used to change the upper bound of an array coordinate axis.

If `schema` is specified, the remaining shape arguments are ignored.

The arguments `upper`, `chunk`, and `overlap` must each have the same number of elements as the dimension of the SciDB array `x`. The `upper` argument should contain double-precision integer values (standard R array index values). The `chunk` and `overlap` arguments should contain integer values.

Optional arguments left unspecified remain unchanged in the output array.

Use `eval=FALSE` to defer expression evaluation for efficiency. Results can be composed to form complex query expressions that can be evaluated in one transaction.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

scidb
Examples

```r
## Not run:
# Add chunk overlap to an array:

x <- as.scidb(iris)
y <- repart(x, overlap=3)

## End(Not run)
```

Description

Change the dimensionality of a SciDB array object.

Usage

```r
## S4 method for signature 'scidb'
reshape(data, schema, shape, dimnames, start, chunks, `eval`=FALSE)
```

Arguments

- `data`: A scidb array object.
- `schema`: Optional character-valued schema.
- `shape`: An integer vector of new dimension sizes.
- `dimnames`: An optional integer vector of new dimension names.
- `start`: An optional integer vector of new dimension starting coordinates.
- `chunks`: An optional integer vector of new dimension chunk sizes.
- `eval`: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Details

If schema is specified the other shape arguments are ignored.

The product of the entries of the shape vector must match the product of the dimensions of data.

The optional dimnames, start and chunks arguments must match the length of shape if they are supplied.

If not specified in either schema or start, the output array will start at the origin.

Value

A scidb reference object.
Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
# Create a 5 x 4 matrix
X <- as.scidb(matrix(rnorm(20),5))

# Reshape into a 3-dimensional 2x2x5 array
Y <- reshape(X, c(2,2,5))

## End(Not run)
```

Description

Return the SciDB array schema.

Usage

`schema(x)`

Arguments

- `x` An object of class `scidb` or `scidbdf`

Value

The SciDB array schema.

Note

This function is nearly equivalent to `x@schema`.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

See Also

`scidb`
Examples

## Not run:

```r
# Upload data to create a 10x10 SciDB matrix
A <- as.scidb(matrix(rnorm(100), 10))
schema(A)
A@schema
```

## End(Not run)

---

**scidb**

Create a scidb reference object.

### Description

Create an array-like R object reference to a SciDB array.

### Usage

```r
scidb(name, gc, `data.frame``)
```

### Arguments

- **name**: Name of the SciDB array to reference.
- **gc**: TRUE means SciDB array shall be removed when R object is garbage collected or R exits. FALSE means SciDB array persists.
- **data.frame**: Return a data.frame-like object (requires 1D SciDB array).

### Details

The referenced array may be any SciDB array. One-dimensional SciDB arrays may be represented as data.frame-like objects in which the SciDB array attributes appear as data.frame columns. Alternatively, 1-d SciDB arrays may be represented as vectors by setting `data.frame`=FALSE.

SciDB arrays of dimension 2 or more appear as R arrays.

Data frame like representations use the `scidbdf` class. N-d array objects and vectors use the `scidb` class.

The `scidb` class supports sparse and dense SciDB arrays of any dimension. Attribute types real, integer (32-bit), logical, and single-character (one byte) are directly supported and may be downloaded to R. Other SciDB attribute types are indirectly supported.

R does not have a native 64-bit integer type. SciDB uses signed 62-bit integer dimensions. The `scidb` package uses R double-precision floating point integers to index SciDB integer dimensions, restricting R to dimension values below $2^{53}$.

With the exception of the empty indexing operation, `[]`, subarray indexing operations return new SciDB reference array objects. Use the empty indexing operation to materialize data from the SciDB backing array into a normal R array.

Sparse SciDB matrices (2-d arrays) are materialized to R as sparse matrices. Higher dimensional sparse arrays are returned as lists of indices and values. See the vignette examples for a more complete discussion of sparsity and various indexing operations.
Value

A `scidb` object that references the indicated SciDB array.

Indexing

The `scidb` and `scidbdf` classes generally follow SciDB database indexing convention, which exhibits some differences with standard R indexing. In particular, note that the starting SciDB integer index is arbitrary, but often zero. The upper-left corner of R arrays is always indexed by [1,1,...]. Subarray indexing operations use the SciDB convention. Thus, zero and negative indices are literally interpreted and passed to SciDB. In particular, negative indices do not indicate index omission, unlike standard R arrays.

Additional indexing notes:

- Use empty brackets, `[]`, to materialize data back to R. Otherwise, indexing operations produce new SciDB array objects.

- Use numeric indices in any dimension in the units of the underlying SciDB array coordinate system. Note that SciDB arrays generally are zero-indexed and may even have negative indices.

- Numeric indexing may include contiguous ranges or vectors of distinct coordinate values, but repeated coordinate values in a single dimension are not allowed. Examples of valid index ranges include `[1:4, c(3,1,5), -10:15]`, but not `[c(1,3,1)]` for example.

- The `scidbdf` class represents 1d SciDB arrays as data frame objects with array attributes as columns. Use either positional numeric or name-based indexing along columns, either with the dollar-sign notation or string indexing. See examples.

- The `scidb` class supports labeled dimension indexing using R `rownames`, `colnames`, or `dimnames` settings. Labels assigned in this way must be provided by 1-d SciDB arrays that map the integer coordinates to character label values. See the examples.

- The `scidb` class supports indexing by other SciDB arrays to achieve the effect of filtering by boolean expressions and similar operations, also illustrated below in the examples section.

- Use the utility between function to avoid forming large sequences to represent huge indexing ranges. For example, use `[between(1,1e9)]` instead of `[1:1e9]`.

- The `diag` function is supported for matrices and vectors.

Indexing by array to select a vector of cell entries is not yet supported.

Author(s)

B. W. Lewis `<blewis@paradigm4.com>`

See Also

`between`, `diag`, `slice`
Examples

```r
## Not run:
scidbconnect()
# A 1-d array representation as a data frame:
data("iris")
A <- as.scidb(iris)
# Inspect the backing SciDB array object details with str:
str(A)

# Subsetting returns a new SciDB array:
A[1:2,]

# Materialize data to R with empty brackets:
A[1:2,][[]]

# Subset data frame-like object columns with 1-based positional or character
# indices. The following are all the same:
A[, "Species"]
A$Species
A[,5]

# Represent the 1-d array as a vector-like object instead:
a <- scidb(A, data.frame=FALSE)
# Interrogate the SciDB array properties with str:
str(a)

# A matrix:
set.seed(1)
X <- as.scidb( matrix(rnorm(20), nrow=5) )
# Diagonal entries of X:
diag(X)
# A sparse matrix with just the diagonal of X:
D = diag(diag(X))
# Materialize this sparse array to R:
D[]

# Produce a sparse matrix of filtered entries and materialize to R:
subset(X, "val > 0")[]

# Short-hand for the same effect:
(X > 0)[]

# Assign row labels to X. Note! We make sure that the index array starts at the
# same starting index as the matrix (zero in this example):
rownames(X) <- as.scidb(data.frame(letters[1:5]),start=0)
# Index by label:
X[c("c","a"), ]

# Filter X by an auxiliary SciDB array condition (we use the rownames array),
# returning the result to R:
X[rownames(X) > "b", ][]
```
# A 3-d array:
X <- build(dim=c(3,2,2), names=c("x","i","j","k"), data="i+j+k")

# A sparse 3-d array filtered with subset:
Y <- subset(X, "x>1")
count(Y)
Y[]

## End(Not run)

<table>
<thead>
<tr>
<th>scidb-class</th>
<th>Class &quot;scidb&quot;</th>
</tr>
</thead>
</table>

**Description**

A class that represents SciDB arrays as R arrays.

**Objects from the Class**

Objects can be created by calls of the form `new("scidb", ...), scidb("ARRAY_NAME", ...), or as.scidb(R_MATRIX, ...)`.  

**Slots**

- name: Object of class "character" scidb array name.  
- attributes: Object of class "character" Vector of SciDB attribute names.  
- dimensions: Object of class "character" Vector of SciDB dimension names.  
- schema: Object of class "character" SciDB array schema.  
- gc: Object of class "environment" An environment used to link the SciDB array to the R garbage collector.  

**S3Class**: Object of class "character" ~

**Methods**

- `%*%` signature(x = "scidb", y = "scidb"): ...
- `%*%` signature(x = "scidb", y = "matrix"): ...
- `%*%` signature(x = "matrix", y = "scidb"): ...
- `%*%` signature(x = "scidb", y = "numeric"): ...
- `%*%` signature(x = "numeric", y = "scidb"): ...
- `%*%` signature(x = "scidb", y = "scidbdf"): ...
- `crossprod` signature(x = "scidb", y = "scidb"): ...
- `tcrossprod` signature(x = "scidb", y = "scidb"): ...
- `crossprod` signature(x = "scidb", y = "missing"): ...
Decimate the n-d SciDB array \( x \) by binning its coordinate system according to the grid argument and applying the aggregate expression on each bin. The grid argument must be the same length as the dimension of the array \( x \). For example if \( x \) is a 10x10 matrix, then \( \text{regrid}(x, \text{c}(2,1)) \) returns a 5x10 matrix replacing the rows with the average of values along every two rows.

Prolong the coordinate system of the n-d SciDB array \( x \) by replicating its values according to the grid parameter. In some cases, \( \text{xgrid} \) can be the inverse operation of \( \text{regrid} \). For example if \( x \) is a 10x10 matrix, then \( \text{xgrid}(x, \text{c}(2,1)) \) returns a 20x10 matrix, replicating values twice along the row dimension.

"Filter" masks elements of a vector for which a predicate (logical) SciDB expression 'f' returns false by making them empty, returning a sparse version of the array.

Display a heatmap-like image of the 2-d scidb array reference object \( x \). \( \text{grid} \) specifies the partitioned array block sizes and \( \text{op} \) is a valid SciDB aggregation function applied to the partitioned chunks.

Return the diagonal of a SciDB matrix as a new 1d SciDB array (a vector).

Return the transpose of the matrix \( x \).

Return a 1-D representation of an array using the SciDB array unpack operator.

Return the diagonal of a SciDB matrix as a new 1d SciDB array (a vector).

Return an array with the sine of each element of the array \( x \). The active attribute is used and should be numeric. The returned array contains a new attribute with "\_sin" appended containing the result. Other trig functions (\( \cos \), \( \tan \), \( \text{asin} \), \( \text{acos} \), \( \text{atan} \)) act similarly.

Return the cosine of each element of the array \( x \). See doc. for \( \text{sin} \).

Return the cosine of each element of the array \( x \). See doc. for \( \text{sin} \).

Return the tangent of each element of the array \( x \). See doc. for \( \text{sin} \).

Return the arcsine of each element of the array \( x \). See doc. for \( \text{sin} \).

Return the arccosine of each element of the array \( x \). See doc. for \( \text{sin} \).

Return the arctangent of each element of the array \( x \). See doc. for \( \text{sin} \).
Notes
SciDB arrays are general n-dimensional sparse arrays with integer dimensions. The scidb class represents SciDB arrays in a way that mimics standard R arrays in many ways. The scidbdf class represents one-dimensional SciDB arrays with one or more attributes that mimic R data frames. Matrix arithmetic operations are overloaded for 2-D numeric arrays.

Author(s)
B. W. Lewis <blewis@paradigm4.com>

See Also
scidb

scidbconnect

Connect to a SciDB database.

Description
Connect to a SciDB database.

Usage
scidbconnect(host=options("scidb.default_shim_host")[[1]],
              port=options("scidb.default_shim_port")[[1]],
              username, password)

Arguments
host The host name or I.P. address of the SciDB database instance to connect to (character).
port The integer port number of the SciDB database simple HTTP service. The default shim ports are 8080 (open) and 8083 (SSL encrypted).
username Optional authentication username (character).
password Optional authentication password (character).

Details
The SciDB connection state is maintained internally to the scidb package. We internalize state to facilitate operations involving scidb objects.
Thus, only one open SciDB connection is supported at a time.
One may connect to and use multiple SciDB databases by sequentially calling scidbconnect between operations. Note that scidb objects are not valid across different SciDB databases.
Use the optional username and password arguments to authenticate the connection with the shim service. Authenticated connections require an encrypted connection with shim, available by default on port 8083.
**Value**

NULL is invisibly returned. SciDB connection errors are propagated to R and may be handled with the usual R error handling mechanisms.

**Note**

Disconnection is automatically handled by the package.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

---

**Description**

A class that represents 1D SciDB arrays similarly to R data.frames.

**Objects from the Class**

Objects can be created by calls of the form `new("scidbdf", ...), scidb("ARRAY_NAME", ...),` or as `scidb(R_DATA_FRAME, ...)`. 

**Slots**

name: Object of class "character" scidb array name.

attributes: Object of class "character" Vector of SciDB attribute names.

dimensions: Object of class "character" Vector of SciDB dimension names.

schema: Object of class "character" SciDB array schema.

gc: Object of class "environment" An environment used to link the SciDB array to the R garbage collector.

.S3Class: Object of class "character" ~

**Methods**

`is.scidbdf` signature(x = "scidbdf"): ...

`is.scidbdf` signature(x = "ANY"): ...

`print` signature(x = "scidbdf"): ...

`head` signature(x = "scidbdf"): ...

`tail` signature(x = "scidbdf"): ...

`show` signature(object = "scidbdf"): ...

`Filter` signature(f="character", x = "scidbdf"): Filter(f,x) 'Filter' masks elements of a vector for which a predicate (logical) SciDB expression 'f' returns false by making them empty, returning a sparse version of the array.
**sin** signature(x = "scidbdf"): \( \sin(x) \) Return an array with the sine of each element of the array \( x \). The first listed attribute is used and should be numeric. The returned array contains a new attribute with "\_sin" appended containing the result. Additional trig functions (\( \cos, \tan, \text{asin}, \text{acos}, \text{atan} \)) act similarly.

**abs** signature(x = "scidbdf"): \( \cos(x) \) See doc. for \( \sin \).

**sqrt** signature(x = "scidbdf"): \( \cos(x) \) See doc. for \( \sin \).

**cos** signature(x = "scidbdf"): \( \cos(x) \) See doc. for \( \sin \).

**tan** signature(x = "scidbdf"): \( \tan(x) \) See doc. for \( \sin \).

**asin** signature(x = "scidbdf"): \( \text{asin}(x) \) See doc. for \( \sin \).

**acos** signature(x = "scidbdf"): \( \text{acos}(x) \) See doc. for \( \sin \).

**atan** signature(x = "scidbdf"): \( \text{atan}(x) \) See doc. for \( \sin \).

**Notes**

Like the related scidb class, the scidbdf class represents SciDB arrays as R objects. The scidbdf class presents 1-D SciDB arrays, potentially with many SciDB attributes (variables), as a data.frame-like object.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

---

**scidbdisconnect**  
*Disconnect from a SciDB database.*

**Description**

SciDB connections are automatically disconnected by the package in normal practice. The scidbdisconnect function forces the current connection to disconnect.

**Usage**

`scidbdisconnect()`

**Value**

NULL is invisibly returned.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>
scidbeval

Description
Evaluate a SciDB expression.

Usage
scidbeval(expr, eval=TRUE, name, gc)

Arguments
expr An R expression that evaluates to a ‘scidb’ or ‘scidbdf’ object.
eval If TRUE, return a ‘scidb’ or ‘scidbdf’ object that has been evaluated by the database and stored to a SciDB array.
name An optional character value that names the stored SciDB array.
gc An optional logical value that, when TRUE, ties the result to the R garbage collector.

Value
Either a scidb or scidbdf object.

Note
Use eval=FALSE to defer expression evaluation for efficiency. Many functions in the SciDB package accept un-evaluated expressions. One can compose these objects to form complex query expressions that can be evaluated in one transaction.

Author(s)
B. W. Lewis<blewis@paradigm4.com>

See Also
scidb

Examples
## Not run:
A <- as.scidb(matrix(rnorm(25),5))
B <- A + A  # Returns a SciDB array promise (not evaluated yet)
C <- scidbeval(A + A)  # Returns a SciDB array that has been evaluated and stored
## End(Not run)
scidblist  

List SciDB database items.

Description

List arrays and various other items available in a SciDB database.

Usage

scidblist(pattern, type = c("arrays", "operators", "functions", "types",  
                         "aggregates", "instances", "queries", "libraries"),  
                         verbose = FALSE, n = Inf)

Arguments

pattern  Filter the list by a regular-expression style pattern.
type     A character string indicating the type of SciDB item to list–defaults to 'array.'
verbose  Include extra available information for some items (arrays, in particular).
n        Maximum number of rows to return.

Value

A data frame of listed items.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
# Not run:
# List information about the SciDB cluster we're connected to:
scidblist('instances')

# List all arrays:
scidblist()

# List all arrays beginning with "A:"
scidblist(pattern="^A")
```

## End(Not run)
Description
Remove (delete) an array from SciDB.

Usage
scidbremove(x, error = warning, async, force, warn=TRUE, recursive=FALSE)

Arguments
x The name of the SciDB array to remove (character), or a vector of array names to remove, or a scidb or scidbdf object.
error Error handling function.
async (Optional) logical: If TRUE, use async option in newer versions of SciDB shim interface to return faster by not waiting for the query to complete. The default value is FALSE.
force (Optional) logical: If TRUE, force removal of specified array, even if the scidb.safe_remove option is set to TRUE.
warn (Optional) logical: Set to FALSE to disable warnings during removal.
recursive (Optional) logical: If true, remove the entire dependency graph of this array from SciDB.

Details
Supply a user-defined error handling function to avoid stopping on error (for example trying to delete a non-existing array). For example, use error=warning to covert errors to warnings but still report them.

The package option scidb.safe_remove adds some protection for accidentally removing arrays. When TRUE (the default value), only temporary arrays associated with the current R session can be removed without also setting the force option. Specify force=TRUE to remove any specified array, or optionally set the options(scidb.safe_remove=FALSE) to disable this safety check.

The async option does not really do anything yet. It will in a future version.

Use recursive=TRUE with great care. It recursively calls scidbremove on the entire dependency graph of an array object.

Value
NULL is invisibly returned. SciDB errors are returned as normal R errors and may be handled accordingly.

Author(s)
B. W. Lewis<blewis@paradigm4.com>
### Examples

```r
## Not run:
data("iris")
x = as.scidb(iris)
# This is the SciDB array name corresponding to x:
print(x$name)

# Let's make an array with a different name:
y = as.scidb(iris, name="iris")
print(y$name)

# Removing everything will only delete arrays whose name begin with
# R_array by default:
scidbremove(scidblist())
scidblist()

# To remove the 'iris' array, use force=TRUE:
scidbremove("iris", force=TRUE)
scidblist()

## End(Not run)
```

---

**scidb_attributes**  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>An object of class scidb or scidbdf</td>
</tr>
</tbody>
</table>

**Value**  
A character vector of SciDB array dimension names.

**Author(s)**  
B. W. Lewis<blewis@paradigm4.com>

**See Also**  
scidb
Examples

```r
## Not run:
# Upload data to create a 10x10 SciDB matrix
A <- as.scidb(matrix(rnorm(100), 10))
scidb_attributes(A)

## End(Not run)
```

Description

Performs Fisher's exact test for testing the null of independence of rows and columns in a contingency table with fixed marginals.

Usage

```r
scidb_fisher.test(a, x="x", m="m", n="n", k="k", alternative="two.sided", 'eval'=FALSE)
```

Arguments

- `a`: a scidb array or scidbdf data frame object.
- `x`: The x value attribute name (see details below).
- `m`: The m marginal value attribute name (see details below).
- `n`: The m marginal value attribute name (see details below).
- `k`: The m marginal value attribute name (see details below).
- `alternative`: indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less".
- `eval`: (Optional) If TRUE, execute the query and store the result array. Otherwise defer evaluation.

Details

For 2 by 2 tables, the null of conditional independence is equivalent to the hypothesis that the odds ratio equals one. "Exact" inference can be based on observing that in general, given all marginal totals fixed, the first element of the contingency table has a non-central hypergeometric distribution with non-centrality parameter given by the odds ratio (Fisher, 1935).

Consider the following 2x2 contingency table:

<table>
<thead>
<tr>
<th></th>
<th>Class I YES</th>
<th>Class I NO</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class II YES</td>
<td>x</td>
<td>a</td>
<td>k = x + a</td>
</tr>
<tr>
<td>Class II NO</td>
<td>b</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td>m = x + b</td>
<td>n = a + c</td>
<td></td>
</tr>
</tbody>
</table>
The \textit{x} input value specifies the name of the SciDB array attribute that indicates the number of 'yes' events in both classifications. The \textit{m} input value specifies the name of the SciDB array attribute that indicates the marginal sum of the first column. The \textit{n} input value specifies the name of the SciDB array attribute that indicates the marginal sum of the second column. The \textit{k} input value specifies the name of the SciDB array attribute that indicates the marginal sum of the first row.

**Value**

A new SciDB array with two new attributes is returned (note that the returned attribute names may be adjusted to account for naming conflicts with existing array attributes):

- \textit{pvalue} the p-value of the test.
- \textit{estimate} an estimate of the odds ratio. Note that the conditional Maximum Likelihood Estimate (MLE) rather than the unconditional MLE (the sample odds ratio) is used.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

**See Also**

- \texttt{scidb phyper qhyper dhyper}

**Examples**

```r
## Not run:
# Create a test array:
a <- scidb("apply(build([x:int64[i=0:0,1,0],2],m,n,k,17])")
scidb_fisher.test(a)[]

# output looks like:
# x m n k pval estimate
# 0 2 12 18 17 0.0005367241 0.4693664

## End(Not run)
```

---

**Description**

Display the package software repository commit log

**Usage**

\begin{verbatim}
show_commit_log()
\end{verbatim}
slice

**Author(s)**
B. W. Lewis <blewis@paradigm4.com>

---

**Description**
Slice a SciDB array along a coordinate value returning a new SciDB array of lower dimension.

**Usage**
slice(x, d, n, `eval`=FALSE)

**Arguments**
- **x**: A scidb object.
- **d**: A character string representation of the coordinate axis to slice along, or logical or numeric index of the axis.
- **n**: The value to slice on.
- **eval**: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

**Value**
A new scidb object of lower dimension.

**Author(s)**
B. W. Lewis <blewis@paradigm4.com>

**See Also**
scidb

**Examples**

```r
## Not run:
# Build a 5x5 matrix:
x <- build("random()
print(x[])

# Slice the 2nd row (i=1 since SciDB matrices start with zero index by default)
y <- slice(x, "i", 1)
print(y[])

## End(Not run)
```
sort-methods

Methods for Function `sort` in Package `scidb`

Description

Aggregate a SciDB array object grouped by a subset of its dimensions and/or attributes.

Usage

```r
## S4 method for signature 'scidb'
sort(x, decreasing=FALSE, attributes, eval)
## S4 method for signature 'scidbdf'
sort(x, decreasing=FALSE, attributes, eval)
```

Arguments

- `x` A `scidb` or `scidbdf` object.
- `decreasing` A logical value: TRUE means sort in decreasing order.
- `attributes` (Required) A character vector of attribute names from array `x` to sort on.
- `eval` (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Details

Create a new array reference object that represents the sorted version of array `x`. The output array reference has the same shape and number of attributes as `x` but the specified attributes appear in sorted order.

Value

A `scidb` or `scidbdf` reference object.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
# Create a copy of the iris data frame in a 1-d SciDB array named "iris."
# Note that SciDB attribute names will be changed to conform to SciDB
# naming convention.
x <- as.scidb(iris,name="iris")

# Sort x by Petal_Width and Petal_Length:
a <- sort(x, attributes=c("Petal_Width","Petal_Length"))
## End(Not run)
```
Str.scidb

Show the structure of a scidb object.

Description
Print a summary of the dimension structure of a scidb array object.

Usage

```r
## S3 method for class 'scidb'
str(object, ...)
```

Arguments

- `object`: A scidb array object.
- `...`: Additional objects for `str` (see `str`).

Value

`str` does not return anything, for efficiency reasons. The obvious side effect is output to the terminal.

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Subarray

Low-level SciDB subarray operator

Description
Return a new SciDB subarray object.

Usage

```r
subarray(x, limits, schema, between=FALSE, `eval`=FALSE)
```

Arguments

- `x`: A scidb object.
- `schema`: An optional SciDB array object or SciDB schema string that defines the coordinate limits.
- `between`: If TRUE, form a SciDB between-style query, otherwise use `subarray`.
- `eval`: If TRUE, execute the query and store the result array. Otherwise defer evaluation.
### Value

A new scidb object.

### Author(s)

B. W. Lewis <blewis@paradigm4.com>

### See Also

scidb

---

**subset-methods**

*Methods for Function subset in Package scidb*

### Description

SciDB filter operations.

### Usage

```r
## S4 method for signature 'scidb'
subset(X, subset, eval)
## S4 method for signature 'scidbdf'
subset(X, subset, eval)
```

### Arguments

- **X**: A scidb or scidbdf object.
- **subset**: A character string representing a SciDB filter expression.
- **eval**: (Optional) If true, execute the query. Otherwise defer evaluation.

### Details

Perform a SciDB filter operation on a SciDB array. The subset filter expression argument can be any valid SciDB filter operation.

### Value

A scidb, scidbdf reference object.

### Note

We use the function name subset because the R function subset closely matches the native functionality of the SciDB operator named filter. The R filter function is an unrelated timeseries function.
substitute

Author(s)

B. W. Lewis <blewis@paradigm4.com>

Examples

### Not run:

```
# Create a copy of the iris data frame in a 1-d SciDB array named "iris."
# Note that SciDB attribute names will be changed to conform to SciDB
# naming convention.
x <- as.scidb(iris, name="iris")

y <- subset(x,"Species='setosa'")
```

### End(Not run)

---

**substitute**

*Replace SciDB NULL values with another value.*

Description

Replace SciDB NULL values with another value.

Usage

```
substitute(x, value, 'attribute', 'eval'=FALSE)
```

Arguments

- `x`: A scidb or scidbdf object.
- `value`: An optional scidb array containing the replacement value.
- `attribute`: An optional attribute to restrict NULL replacement to.
- `eval`: (Optional) If true, execute the query and store the result array. Otherwise defer evaluation.

Details

If `value` is missing, a value of numeric zero, false, or the string "0" will be used to replace NULL values depending on the replacement attribute type.

If `attribute` is missing, then an attempt will be made to replace all NULL values in the array.

Value

A new scidb or scidbdf object.

Author(s)

B. W. Lewis <blewis@paradigm4.com>
See Also

scidb

Examples

## Not run:
x <- scidb("build<v:double null>[i=1:3,3,0],'(1)[(null),(null),(null)],true")
y <- substitute(x)

## End(Not run)

---

`summary.scidb`  
*Summarize scidb object class.*

**Description**

Summarize scidb object class. This is not a really useful function yet.

**Usage**

```r
## S3 method for class 'scidb'
summary(x)
```

**Arguments**

- `x`  
a scidb array object.

**Author(s)**

B. W. Lewis <blewis@paradigm4.com>

---

`svd`

**Description**

Compute a partial or full singular value decomposition of a SciDB matrix.

**Usage**

```r
## S4 method for signature 'scidb'
svd(x, nu, nv)
```
Arguments

x A scidb object representing a matrix.

nu Optional number of singular values/vectors to compute. See notes below.

nv Not used.

Value

A list of three scidb objects corresponding to a matrix of left singular vectors, a vector of singular values, and a matrix of transposed right singular vectors, respectively.

Note

If the nu argument is specified, the SciDB tsvd function will be used and the matrix x may be dense or sparse. Otherwise the full SVD is computed by SciDB and the matrix x must be dense. The nv argument is not used and appears in the function signature only for compatibility with the base svd function. This function always sets nv = nu.

Author(s)

B. W. Lewis<blewis@paradigm4.com>

See Also

scidb

Examples

```r
## Not run:
x <- as.scidb(matrix(rnorm(500*500),500))
y <- svd(x)
print(y)

## End(Not run)
```

Methods for Function sweep in Package scidb

Description

Return an array obtained from an input array by sweeping out a summary statistic.

Usage

```r
## S4 method for signature 'scidb'
sweep(x, MARGIN, STATS, FUN = "-", eval = FALSE, name)
```
Arguments

- **x**: a scidb object.
- **MARGIN**: a vector of indices giving the extent(s) of 'x' which correspond to 'STATS'.
- **STATS**: the summary statistic which is to be swept out.
- **FUN**: the function to be used to carry out the sweep. (Default is ".".)
- **eval**: (Optional) If TRUE, execute the query and store the result array. Otherwise defer evaluation.
- **name**: If x is a multi-attribute array, specify the attribute to work on, default to the first listed array attribute.

Details

- **STATS** must be a vector whose length matches the dimension size of the specified **MARGIN**. Note that this is a more restrictive condition than the usual R **sweep** function.
  - The default value of **FUN** is ".". **FUN** must be a quoted binary operator.

Value

- A scidb object.

Author(s)

- B. W. Lewis <blewis@paradigm4.com>

Examples

```r
## Not run:
set.seed(1)
A <- as.scidb(matrix(rnorm(20),nrow=5))
# Center the matrix
B <- sweep(A, MARGIN=2, STATS=apply(A,2,mean))
# Let's see:
apply(B,2,mean)[]
## End(Not run)
```
Index

*Topic classes
  scidb-class, 54
  scidbdf-class, 57
*Topic distribution
  hypergeometric-methods, 29
*Topic package
  scidb-package, 3
  %%%, matrix, scidb-method (scidb-class), 54
  %%%, numeric, scidb-method (scidb-class), 54
  %%%, scidb, matrix-method (scidb-class), 54
  %%%, scidb, numeric-method (scidb-class), 54
  %%%, scidb, scidb-method (scidb-class), 54
  %%%, scidb, scidbdf-method (scidb-class), 54
  %%%,(comparison-methods), 16
  %%%, scidb-method (comparison-methods), 16
  %%%, scidbdf-method (comparison-methods), 16
  %%%, (comparison-methods), 16
  %%%, scidb-method (comparison-methods), 16
  %%%, scidbdf-method (comparison-methods), 16
  %%%, (comparison-methods), 16
  %%%, scidb-method (comparison-methods), 16
  %%%, scidbdf-method (comparison-methods), 16
  %%%, (comparison-methods), 16
  %%%, scidb-method (comparison-methods), 16
  %%%, scidbdf-method (comparison-methods), 16
  %%%, (comparison-methods), 16
  abs, scidb-method (scidb-class), 54
  abs, scidbdf-method (scidbdf-class), 57
  acos, scidb-method (scidb-class), 54
  acos, scidbdf-method (scidbdf-class), 57
  aggregate, 18
  aggregate (aggregate-methods), 3
  aggregate, scidb-method (aggregate-methods), 3
  aggregate, scidbdf-method (aggregate-methods), 3
  all.equal, 6
  antijoin, 7
  apply, scidb-method (scidb-class), 54
  as.scidb, 8
  asin, scidb-method (scidb-class), 54
  asin, scidbdf-method (scidbdf-class), 57
  atan, scidb-method (scidb-class), 54
  atan, scidbdf-method (scidbdf-class), 57
  attribute_rename, 9, 15, 22
  between, 10, 52
  bind, 11
  build, 12
  c (c-methods), 14
  c, scidbdf-method (c-methods), 14
  c-methods, 14
  cast, 10, 15, 22
  colnames, 23
  comparison-methods, 16
  cos, scidb-method (scidb-class), 54
  cos, scidbdf-method (scidbdf-class), 57
  count (grand-methods), 26
  count, scidb-method (grand-methods), 26

73
count, scidbdf-method (grand-methods), 26
crossprod, scidb, missing-method
   (scidb-class), 54
crossprod, scidb, scidb-method
   (scidb-class), 54
cumulate, 18
df2scidb, 8, 9, 19
dhyper, 64
diag, 52
diag, scidb-method (scidb-class), 54
diff, scidb, 20
dim, scidb, 21
dimension_rename, 10, 15, 22
dimensions, 21
dimnames, scidb, 23
Filter (scidb-class), 54
Filter, character, scidb-method
   (scidb-class), 54
Filter, character, scidbdf-method
   (scidbdf-class), 57
glm.fit, 24
glm.fit, scidb, ANY, MNSN-method
   (glm.fit), 24
glm.fit, scidb-method (glm.fit), 24
glm.scidb, 42
glm.scidb (glm.fit), 24
grand-methods, 26
head, scidb-method (scidb-class), 54
head, scidbdf-method (scidbdf-class), 57
hist, hist-methods, 28
hist, scidb-method (hist-methods), 28
hist, scidbdf-method (hist-methods), 28
hist-methods, 28
hypergeometric-methods, 29
image (scidb-class), 54
image, scidb-method (scidb-class), 54
index_lookup, 30
iquery, 31
is.scidb, 33
is.scidb, ANY-method (scidb-class), 54
is.scidbdf, 33
is.scidbdf, ANY-method (scidbdf-class), 57
is.scidbdf, scidbdf-method
   (scidbdf-class), 57
lag, 34
lag, scidb-method (lag), 34
length.scidb, 35
max (grand-methods), 26
max, scidb-method (grand-methods), 26
max, scidbdf-method (grand-methods), 26
mean (grand-methods), 26
mean, scidb-method (grand-methods), 26
mean, scidbdf-method (grand-methods), 26
median (grand-methods), 26
median, scidb-method (grand-methods), 26
median, scidbdf-method (grand-methods), 26
merge (merge-methods), 35
merge, scidb-method (merge-methods), 35
merge, scidbdf-method (merge-methods), 35
merge-methods, 35
min (grand-methods), 26
min, scidb-method (grand-methods), 26
min, scidbdf-method (grand-methods), 26
model_scidb (glm.fit), 24
na.locf (na.locf-methods), 37
na.locf, scidb-method (na.locf-methods), 37
na.locf, scidbdf-method
   (na.locf-methods), 37
na.locf-methods, 37
names, scidb, 38
ncol.scidb (nrow.scidb), 39
nrow.scidb, 39
Ops.scidb, 39
persist, 40
phyper, 64
phyper, hypergeometric-methods, 29
phyper, scidb_or_scidbdf-method
   (hypergeometric-methods), 29
predict.glm.scidb, 26, 41
print, scidb-method (scidb-class), 54
print, scidbdf-method (scidbdf-class), 57
print.scidb, 43
project, 43
qhyper, 64
qhyper, hypergeometric-methods, 29
qhyper, scidb_or_scidbdf-method
   (hypergeometric-methods), 29
INDEX

quantile, 44, 45
redimension, 45
regrid(scidb-class), 54
regrid, scidb-method (scidb-class), 54
rename, 47
repart, 48
reshape (reshape-methods), 49
reshape, scidb-method (reshape-methods), 49
reshape, scidb-df-method (reshape-methods), 49
reshape-methods, 49
rownames, 23
schema, 50
scidb, 6, 7, 9-13, 15, 22, 26, 31, 34, 38, 40,
42, 44, 47, 48, 50, 51, 56, 59, 62, 64,
65, 68, 70, 71
scidb-class, 54
scidb-package, 3
scidb-attributes, 62
scidb_fisher_test, 63
scidbconnect, 56
scidbdf-class, 57
scidbdisconnect, 58
scidbeval, 59
scidbls, 60
scidbls (scidbls), 60
scidbremove, 61
scidb_rmv (scidbremove), 61
sd (grand-methods), 26
sd, scidb-method (grand-methods), 26
sd, scidb-df-method (grand-methods), 26
show, scidb-method (scidb-class), 54
show, scidb-df-method (scidbdf-class), 57
show_commit_log, 64
sin, scidb-method (scidb-class), 54
sin, scidb-df-method (scidbdf-class), 57
slice, 52, 65
sort (sort-methods), 66
sort, scidb-method (sort-methods), 66
sort, scidb-df-method (sort-methods), 66
sort-methods, 66
sqrt, scidb-method (scidb-class), 54
sqrt, scidb-df-method (scidbdf-class), 57
str, 67
str.scidb, 67
subarray, 67
subset (subset-methods), 68
subset, scidb-method (subset-methods), 68
subset, scidb-df-method (subset-methods), 68
subset-methods, 68
substitute, 69
sum (grand-methods), 26
sum, scidb-method (grand-methods), 26
sum, scidb-df-method (grand-methods), 26
summary.scidb, 70
svd, 70
svd, scidb-method (svd), 70
sweep (sweep-methods), 71
sweep, scidb-method (sweep-methods), 71
sweep-methods, 71
t, scidb-method (scidb-class), 54
tail, scidb-method (scidb-class), 54
tail, scidb-df-method (scidbdf-class), 57
tan, scidb-method (scidb-class), 57
tan, scidb-df-method (scidbdf-class), 57
tcrossprod, scidb, missing-method (scidb-class), 54
tcrossprod, scidb, scidb-method (scidb-class), 54
unpack, scidb-method (scidb-class), 54
var (grand-methods), 26
var, scidb-method (grand-methods), 26
var, scidb-df-method (grand-methods), 26
xgrid (scidb-class), 54
xgrid, scidb-method (scidb-class), 54