Package 'externalVector'

September 24, 2012

Title Vector objects for R with external storage

Version 1.22.0

Author Saikat DebRoy

Description Basic class definitions and generics for external pointer
based vector objects for R. A simple in-memory implementation is also provided.

Collate init.R allocators.R index.R storage.R simpleStorage.R
primitivevectors.R externalVector.R coerce.R matrixIndex.R
indirect.R Groups.R basicFuns.R zzz.R

Depends R (>= 1.8.0), methods, stats

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

License LGPL

LazyLoad yes

biocViews Infrastructure

R topics documented:

allIndex-class . . . 2
allNA . . . 3
allocators . . . 4
as.Rvector . . . 5
asEach . . . 6
defaultElement . . . 7
exprMatrix-class . . . 7
externalAllocator-class . . . 8
externalMatrix . . . 9
externalResource-class . . . 10
externalStorage-class . . . 12
externalVector . . . 13
externalVector-class . . . 14
externalVectorWithStorage-class . . . 16
gcAllocator-class . . . 17
getExternalStorage . . . 19
getNativeStorageMethods . . . 19

indirectExternalMatrix-class 20
indirectExternalVector-class 21
internalType 23
makeSlice 23
matrixIndex-class 24
NAIndex-class 25
nativeStorageMethods-class 26
nativeStorageMethodsList-class 28
noneIndex-class 29
nonSlotAttributes 30
positiveIndex-class 31
rebind 32
setDimNames 32
setExternalStorageClass 33
setVirtualMethod 34
simpleStorage-class 36
sliceIndex-class 37
SubscriptList 38
Subset 39
union-class 39
var 39
vectorIndex-class 41

Index **42**

`allIndex-class` *Class "allIndex", index class representing a subscript for all elements of a vector or an array dimension*

Description

When used as a subscript for a vector object `x`, this is equivalent to `seq(length(x))`. When used as a subscript for the `i`-th dimension of an array object `x`, this is equivalent to `seq(dim(x)[i])`.

Objects from the Class

Objects can be created by calls of the form `new("allIndex")`. Objects from this class can not be modified and they are all equivalent.

Slots

`.Data`**:** Object of class `"logical"`, this is always equal to `TRUE`.

Extends

Class `"logical"`, directly. Class `"vectorIndex"`, directly.

Methods

Signature components for the methods are:

x	The class "allIndex"
scalar	Length one positive "integer"
i	The class "ANY"
value	The class "ANY"
.Object	The class "allIndex"

`x+scalar`**:** Add `scalar` to `x`. Result is `x`.

`scalar+x`**:** Add `scalar` to `x`. Result is `x`.

`x*scalar`**:** Multiply `x` by `scalar`. Result is `x`.

`scalar*x`**:** Multiply `x` by `scalar`. Result is `x`.

dim(x) <- value Set a dimension on `x`. If value is of length 2, this may create a "matrixIndex".

`x[i]`**:** Subset x. Result is i.

`x[]`**:** Subset x with the subscript missing. Result is x.

`x[i] <- value`**:** Subassign x. Results in an error (as objects from class `"allIndex"` are not mutable.

`x[[i]] <- value`**:** Subassign x. Results in an error (as objects from class `"allIndex"` are not mutable.

`allNA(x)`**:** Are all elements of `x` `NA`? Result is always `FALSE`.

`anyNA(x)`**:** Is any element of `x` `NA`? Result is always `FALSE`.

`initialize(.Object)`**:** Initialize an object from class `"allIndex"`.

`length(x)`**:** Return the length of `x` which is always `Inf`.

`length(x) <- value`**:** Modify the length of `x`. Always results in an error.

See Also

`vectorIndex-class` for the super class of `"allIndex"`.

`allNA` *The generics "allNA" and "anyNA".*

Description

The two generic `"allNA"` returns `TRUE` if all of its arguments are NA's and `FALSE` otherwise.

The two generic `"anyNA"` returns `TRUE` if any of its arguments is NA and `FALSE` otherwise.

Usage

```
allNA(x)
anyNA(x)
```

Arguments

`x`	An object on which `is.na` can be applied.

Value

Either TRUE or FALSE.

Examples

```
allNA(1:3)
allNA(c(NA, NA, NA))
allNA(c(1, 2, NA))
anyNA(1:3)
anyNA(c(NA, NA, NA))
anyNA(c(1, 2, NA))
```

---

allocators *Generics associated with external resources and their allocators*

Description

These are generics used with objects of class "externalResource" and "externalAllocator".

Usage

```
allocator(resource)
getPointer(resource)
initializeResource(resource, ptr, size, type, ...)
allocatedSize(resource)
allocatedType(resource)
allocate(resource, alloc, size, type, ...)
deallocate(resource, alloc)
external.size(resource, alloc)
external.size(resource, copy, alloc) <- value
reinitializePointer(resource, alloc)
```

Arguments

resource	An object which is an "externalResource".
alloc	An object which is an "externalAllocator".
ptr	An external pointer.
size	The size - if given for a non-vector type, the numebr of bytes. Otherwise, the length of the vector.
type	Object represnting the type stored in the resource.
copy	Logical, if TRUE (the default), then the new memory is initialized to the content of the old memory for the minimum of old and new sizes. Content of any uninitialized memory is undefined. Under any circumstance, the inherent type of the allocated memory remains the same as its initial value.
value	Integer, new size.
...	Further arguments passed to or from other functions.

Value

`allocator` returns the allocator to be used by default with `resource`.

`getPointer` returns the `"externalptr"` associated with `resource`.

`initializeResource` intializes `resource` with `ptr` of type `"externalptr"` and returns `resource`.

`allocatedSize` returns the size of memory to be allocated for `resource`. If `allocatedType(resource)` is an R basic vector type, then the size is the length of the vector. Otherwise the size is the total number of bytes.

`allocatedType` returns an object representing the type to be stored in the resource

`allocate` allocates the external pointer in resource using the allocator `alloc` for `resource`. If `type` is a basic vector object, then it allocates an object of same mode with length `size` and otherwise allocates `size` bytes of raw memory. The `resource` object is initialized by a call to `initializeResource`.

`deallocate` asks the allocator `alloc` to deallocate the memory in `resource`. The result is allocator dependant.

`external.size` returns the argument `size` used in the last call to `allocate` for `resource`.

The replacement form can be used to change the size of the allocated memory. If `value` is same as `external.size(resource)`, then no action is taken. Otherwise, this reallocates the memory in `resource` using the allocator `alloc` (or `allocator(resource)` if `alloc` is missing) with new size `value` and the same type as earlier.

`reinitializePointer` tries to reinitialize the memory pointer in resource after `resource` was saved and restored as an R image (by serialization code, by saving the R workspace, or by an explicit call to `save`).

See Also

`externalAllocator-class`, `externalResource-class`

---

as.Rvector *convert an object to an R basic vector or matrix*

Description

`as.Rvector`, a generic, attempts to coerce its argument into a basic R vector of a convenient mode. All attributes of `x` are preserved.

Default methods are provided for basic R objects which are returned unchanged by as.Rvector.

`as.Rmatrix`, a generic, first converts its argument to a basic R vector and then applies as.matrix to the result.

Usage

```
as.Rvector(x)
as.Rmatrix(x)
```

Arguments

`x`	An object

Details

This preserves all the non-slot attributes of the object `x` when converting to a basic R vector. In addition, if any of `dim(x)`, `dimnames(x)`, `names(x)` is not `NULL`, the corresponding attribute in the result is also set.

Value

The value from as.Rvector is a basic R vector of mode "logical", "integer", "numeric", "complex", "character" or "list". The value from as.Rmatrix is a basic R matrix.

See Also

`as.vector` for converting an object to an appropriate vector with all attributes removed and `as.matrix` for converting a vector like object to a matrix.

Examples

```
x <- 1:2
names(x) <- letters[1:2]
as.vector(x)
as.Rvector(x)
as.Rmatrix(x)
```

---

asEach *Coerce each element of an external vector to a particular type*

Description

This generic coerces each element of an external vector to a particular type.

Usage

```
asEach(x, type, arrayOnly = FALSE)
```

Arguments

`x`	An external vector object.
`type`	Character-string, a basic vector mode.
`arrayOnly`	If arrayOnly is `FALSE` (the default) do not copy extra slots like `Names`, `DimNames` or `Dim`.

Value

Another external vector object of same length but with `class(defaultElement(x)) == class(type)`.

See Also

`as` to convert an object to a different class.

Examples

```
x <- externalCharacter(4)
x[] <- c("1", "2", "3", "4")
asEach(x, "integer")
```

---

defaultElement *Return the default element for a vector object.*

Description

For a vector object, this generic returns a length one vector with the default element for the vector as the first (and only) element.

Usage

```
defaultElement(x)
```

Arguments

x	A vector object.

Value

A length one vector with the default element.

See Also

externalVector-class for the method for this generic for the "externalVector" class.

Examples

```
defaultElement(1:5)
defaultElement(list(1, 2))
```

---

exprMatrix-class *Class union for objects behaving like a matrix*

Description

Class union "exprMatrix" represents the type of exprs and se.exprs slots in "exprSet" objects. One of the classes in the union is always "matrix"

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "exprMatrix" in the signature.

`externalAllocator-class`

Class "externalAllocator", base class for memory allocators for external resources

Description

Class `"externalAllocator"` is a virtual class with no slots. It represents memory allocators that allocate raw memory to be held in an object of class `"externalptr"`. Instead of returning the `"externalptr"` object directly, the allocators deal with subclasses of `"externalResource"` that hold an object of class `"externalptr"`.

Objects from the Class

A virtual Class: No objects may be created from it.

Virtual Methods

Attempt to execute these methods would result in an error unless they have been redefined for a subclass of `"externalAllocator"`.

Signature components for implementation of the methods:

resource	The class "externalResource"
alloc	A subclass of "externalAllocator"
size	The class "ANY"
type	The class "ANY"
copy	The class "logical"
value	The class "ANY"

Description of the virtual methods:

allocate(resource, alloc, size, type, ...): Allocate the external pointer in `resource` using the allocator `alloc`. If `type` is a basic vector object, then allocate an object of same mode with length `size`. Otherwise allocate `size` bytes of raw memory. This method should end with a call to `initializeResource` to initialize the `resource` object.

deallocate(resource, alloc): The allocator `alloc` should try to dealloacte the raw memory in `resource`. It should not modify the `resource` object in any way other than modifying the object of class `"externalptr"` in `resource` to reflect the deallocation.

external.size Return the size of the allocated memory in `resource`.

external.size<-(resource, copy, alloc, value): If `value` is same as `external.size(resource)`, then no action is taken. Otherwise, reallocate the memory in `resource` using the allocator `alloc` with new size `value`. If `copy` is `TRUE` (the default), then the new memory is initialized to the content of the old memory for the minimum of old and new sizes. Content of any uninitialized memory is undefined.

reinitializePointer(resource, alloc): If the object `resource` was saved as an R image (by serialization code, by saving the R workspace, or by an explicit call to `save`) then the raw memory pointer in any `"externalptr"` object in it would be set to `0`. This method tries to reinitialize the raw memory pointer. The exact result is allocator dependant.

Author(s)

Saikat DebRoy <<saikat@stat.wisc.edu>>

See Also

externalResource-class for more on how to use an allocator with objects from subclasses of "externalResource".

gcAllocator-class for an example of a subclass of "externalAllocator".

setVirtualMethod for more on virtual methods.

---

externalMatrix *Create an external matrix object*

Description

This function can be used to create a new external matrix object.

Usage

```
externalMatrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames =
NULL, storageClass = (if (is(data, "externalVectorWithStorage"))
class(data@storage) else "simpleStorage"), indirect = FALSE)
```

Arguments

data	An optional data vector.
nrow	The desired number of rows.
ncol	The desired number of columns.
byrow	Not used.
dimnames	A dimnames value for the external matrix: a list of length 2.
storageClass	name of the subclass of "externalStorage" to be used for storing the external vector elements.
indirect	If TRUE, return an "indirectExternalVector" object.

Details

This function is similar to the matrix function in R base package.

Value

If indirect is FALSE (the default) a new external matrix object with correct dimension and dimnames. Otherwise create a new external vector object but return it by wrapping it in an "indirectExternalMatrix" object with the correct dimension and dimnames.

See Also

matrix

Examples

```
x <- externalMatrix(1:6, nrow=2, ncol=3)
dim(x)
x[1, 1:2] # drop = FALSE by default
log(x)
```

`externalResource-class`

Class "externalResource", base class for external resource allocated by an externalAllocator

---

Description

Class `"externalResource"` is a virtual class with no slots. External allocators, represented by subclasses of class `"externalAllocator"` can only allocate external pointers contained in objects from a subclass of `"externalResource"`.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods for the virtual class "externalResource"

These methods are defined for the virtual class "externalResource" and need not be redefined by its subclasses.

Signature components for the methods are:

resource	The class "externalResource"
alloc	The class "missing"
ptr	The class "externalptr"
size	The class "ANY"
type	The class "ANY"
copy	The class "logical"
value	The class "ANY"

The argument `alloc` has the class "missing" wherever it appears in these methods. That means, to invoke these methods, the argument `alloc` must be ommitted from the call and to use any of the additional arguments, they must be specfied by name. For example, use `allocate(resource, size=size, type=type)` and not `allocate(resource, size, type)`.

Description of the methods:

allocatedType(resource): Returns `NULL`.

allocate(resource, alloc, size, type, ...): Allocate the external pointer in resource using the default allocator for `resource`. If `type` is a basic vector object, then allocate an object of same mode with length `size`. Otherwise allocate `size` bytes of raw memory. The `resource` object is initialized by a call to `initializeResource`.

deallocate(resource, alloc): Ask the default allocator to deallocate the memory in `resource`. The result is allocator dependant.

external.size Return the argument `size` used in the last call to `allocate` for `resource`

external.size<-(resource, copy, alloc, value): If `value` is same as `external.size(resource)`, then no action is taken. Otherwise, reallocate the memory in `resource` using the default allocator with new size `value` and the same type as earlier. If `copy` is `TRUE` (the default), then the new memory is initialized to the content of the old memory for the minimum of old and new sizes. Content of any uninitialized memory is undefined. Under any circumstance, the inherent type of the allocated memory remains the same as its initial value.

initialize(.Object): Code called by new("resourceSubclass", ...) if "resourceSubclass" is a subclass of "externalResource" and either has no `initialize` method of its own or its `initialize` method has `callNextMethod()` in its body. Returns the result of `allocate(.Object, ...)`

reinitializePointer(resource, alloc): If the object `resource` was saved as an R image (by serialization code, by saving the R workspace, or by an explicit call to `save`) then the raw memory pointer in any `"externalptr"` object in it would be set to `0`. This method tries to reinitialize the raw memory pointer. The exact result is allocator dependant.

Virtual Methods

Attempt to execute these methods would result in an error unless they have been redefined for a subclass of `"externalResource"`.

Signature components for implementation of the methods:

resource	A subclass of "externalResource"
ptr	The class "externalptr"
size	The class "ANY"
type	The class "ANY"

Description of the virtual methods:

allocatedSize(resource): Size of memory to be allocated for `resource`. If `allocatedType(resource)` is an R basic vector type, then the size is the length of the vector. Otherwise the size is the total number of bytes.

allocator(resource): The allocator to be used by default with this `resource`.

getPointer(reource): Return the `"externalptr"` associated with this `resource`.

initializeResource(resource, ptr, size, type, ...): Intialize `resource` with `ptr` of type `"externalptr"`. The `size` and `type` arguments are identical to that obtained from previous calls to `allocatedSize(resource)` and `allocatedType(resource)`.

Other Virtual Methods

These methods must be redefined for subclasses for `"externalAllocator"`. It is not necessary to define them for specific subclasses of `"externalResource"`.

Signature components for implementation of the methods:

resource	The class "externalResource"
alloc	A subclass of "externalAllocator"
size	The class "ANY"
type	The class "ANY"
copy	The class "logical"
value	The class "ANY"

Description of the virtual methods:

allocate(resource, alloc, size, type, ...): Allocate the external pointer in `resource` using the allocator `alloc`.

deallocate(resource, alloc): Ask the allocator `alloc` to deallocate the memory in `resource`.

external.size Return the size of the allocated memory in `resource`.

external.size<-(resource, copy, alloc, value): Reallocate the memory in `resource` using the allocator `alloc`.

reinitializePointer(resource, alloc): If the raw memory pointer in `resource` is zero, try to reinitialize it.

Author(s)

Saikat DebRoy <<saikat@stat.wisc.edu>>

See Also

`externalAllocator-class` for more details on how to use an allocator with objects from subclasses of `"externalResource"`.

`gcAllocator-class` for an example of a simple subclass of `"externalResource"`.

`setVirtualMethod` for more on virtual methods.

---

externalStorage-class *Class "externalStorage", base class for external storage for objects of class "externalVectorWithStorage"*

Description

The `"externalStorage"` class represents the base class of the storage backend implementation used by the class `"externalVectorWithSToraghe"`. A pointer to the external storage allocated or accessed by an object of class `"externalStorage"` is kept in an external pointer.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`ptr`**:** Object of class `"externalptr"`, keeps the pointer to the external storage.

`type`**:** Object of class `"vector"`, a vector object of length one representing the type of object stored in the external storage. Usually one of the basic R vector types.

`length`**:** Object of class `"integer"`, Cached value for the length of the vector stored in the external storage.

`methodsptr`**:** Object of class `"externalptr"`.

Extends

Class `"externalResource"`, directly.

Methods

Signature components for the methods:

resource	The class "externalStorage"
x	The class "externalStorage"
ptr	The class "externalptr"
size	The class "ANY"
type	The class "ANY"

allocator(resource): The allocator to be used by default with this `resource`. Returns an object of class `"gcAllocator"`.

getPointer(reource): Return the `ptr` slot.

initializeResource(resource, ptr, size, type, ...): Set the `ptr` slot of resource to `ptr`.

initialize(.Object, type = logical(1), length = as.integer(0)): This checks the validity of `type` and `length` arguments and coerce the `length` argument to class `"integer"` before using them to set the corresponding slots. Finally a call to `callNextMethod` is made to use the initializer for class `"externalResource"` which, via a call to the `initializeResource` method above sets `ptr` slot.

internalType(x): The type of object to be stored within the resource. It is an R basic vector type (`"logical"`, `"integer"`, `"numeric"`, `"complex"`, `"character"` or `"list"`).

getNativeStorageMethods(x): Get the `"externalptr"` stored in the `"nativeStorageMethods"` object associated with a particular subclass of `"externalStorage"` and a basic R vector type.

See Also

`setExternalStorageClass` for how to set a subclass of `"externalStorage"`.

`nativeStorageMethods-class` for the C structure containing C function pointers for a particular storage methods class.

`externalVectorWithStorage-class` for how the `"externalStorage"` class is used.

`simpleStorage-class` for a simple implementation of `"externalStorage"`.

`externalVector` *Creating external vectors*

Description

These functions can be used to create new external vector objects with newly allocated storage.

Usage

```
externalVector(type = "logical", length = 0, storageClass = "simpleStorage", indirect = FALSE)
externalLogical(length = 0, storageClass = "simpleStorage", indirect = FALSE)
externalInteger(length = 0, storageClass = "simpleStorage", indirect = FALSE)
externalNumeric(length = 0, storageClass = "simpleStorage", indirect = FALSE)
externalComplex(length = 0, storageClass = "simpleStorage", indirect = FALSE)
externalCharacter(length = 0, storageClass = "simpleStorage", indirect = FALSE)
```

Arguments

type	A character-string, the name of the class of basic R vectors to be represented by the new external vector object.
length	Length of the new external vector
storageClass	name of the subclass of "externalStorage" to be used for storing the external vector elements.
indirect	If TRUE, return an "indirectExternalVector" object.

Details

These functions are similar to the vector, logical, integer, numeric, complex and character functions in the R base package. These functions have an extra argument storageClass to specify the external storage class to use. The default is to use "simpleStorage" which uses garbage collected memory from R for storage.

Value

If indirect is FALSE (the default) the a new external vector object of the given length, given type with elements stored in an external storage object with class storageClass. Otherwise create the new external vector object but return it by wrapping it in an "indirectExternalVector" object.

See Also

vector, logical, integer, numeric, complex, character, list

Examples

```
x <- externalVector("numeric", 4)
x[] <- 1:4
x[1:3]
y <- x+2
y
x+y
```

---

externalVector-class *Class "externalVector", base class for vector objects stored in an external resource*

Description

This class represents objects that behave like R basic vectors but are stored in some external resource.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

Common signature components for the methods:

x	The class "externalVector"
X	The class "externalVector"

Description of the methods:

asEach(x, type, arrayOnly) Coerce `x` so that it now behaves like a basic R vector with the same class as `type`.

c(x, ..., recursive) Combine all the arguments into a single vector. Behaves like a basic R vector with the same class as `type`.

as.Rvector(x) Coerce `x` to a basic R vector with same names, dimension and dimnames (if any of these are present in `x`).

as.vector(x, mode) Return an external vector which behaves like a basic R vector of mode `mode`.

as.logical(x, ...) Return an external vector with all elements coerced to logical.

as.integer(x, ...) Return an external vector with all elements coerced to integer.

as.double(x, ...) Return an external vector with all elements coerced to numeric.

as.numeric(x, ...) Return an external vector with all elements coerced to numeric.

as.charcater(x) Return an external vector with all elements coerced to character.

as.list(x, ...) Return an external vector that behaves like a list.

as.single(x, ...) Return an external vector with all elements coerced to numeric and a "CSingle" attribute set to `TRUE`.

as.data.frame(x, row.names, optional) Create a data.frame object from `x`.

as.matrix(x) If `length(dim(x))` is 2, return `x` unchanged. Otherwise, return `x` as a matrix with a single column.

defaultElement(x) The default element of `x` contained in a basic R vector.

dimnames(x) <- value Set dimnames of `x` to value.

names(x) <- value Set names of `x` to value.

show(object) Display `object`.

apply(X, MARGIN, FUN, ...) Coerce X to an R matrix and call the default `apply` method.

lapply(X, FUN, ...) Lapply for "externalVector" class.

sapply(X, FUN, ..., simplify, USE.NAMES) Sapply for "externalVector" class.

mean(x, trim = 0, na.rm = FALSE, ...) Mean and trimmed mean of `x`.

median(x, na.rm = FALSE) Mean of `x`.

Virtual Methods

Common signature components for implementation of the virtual methods:

x	A subclass of "externalVector"
value	The class "ANY"

Description of the virtual methods:

setDimNames(x, value) Set the dimnames of `x` to `value` without any error checking or coercion. Return the modified `x`.

setNames(x, value) Set the names of `x` to `value` without any error checking or coercion. Return the modified `x`.

See Also

externalVectorWithStorage-class,matrixIndex-class, indirectExternalVector-class, indirectExternalMa
for subclasses of "externalVector".

---

externalVectorWithStorage-class

Class "externalVectorWithStorage", external vector class with an external storage for the vector

---

Description

This is a subclass of "externalVector" that implements the use of arbitrary external backends for storing the elements of the vector.

Objects from the Class

Objects can be created by calls of the form externalVector(type, lengtyh, storageClass).

Slots

storage: Object of class "externalStorage", the storage backend for the vector elements.

Names: Object of class "vectorNamesType", names of the elements.

Dim: Object of class "integerOrNull", dimension of the object.

DimNames: Object of class "listOrNull", dimnames of the object

Extends

Class "externalVector", directly. Class "vectorNamesType", directly.

Methods

e1 op e2 For any arithmetic or comparison operation op this returns the result of the operation. Either (or both) of e1 and e2 are of class "externalVectorWithSTorage".

Complex(z) Return the result of an operation in the Complex group generic.

Math(x) Return the result of an operation in the Math group generic.

Summary(x) Return the result of an operation in the Summary group generic.

log(x) Return natural logarithm of x.

log(x, base) Return logarithm of x with base base.

x[i, j, . . . , drop=FALSE] Extract a subset of x according to the given subscripts.

x[i, j, . . . <- value] Subassign of x by value according to the given subscripts.

as.Rvector(x) Coerce x to a basic R vector with same names, dimension and dimnames (if any of these are present in x).

asEach(x, type, arrayOnly) Coerce x so that it now behaves like a basic R vector with the same class as type.

colSums(x, na.rm, dims) Return the column sums of x.

colMeans(x, na.rm, dims) Return the column means of x.

rowSums(x, na.rm, dims) Return the row sums of `x`.

rowMeans(x, na.rm, dims) Return the row means of `x`.

dim(x) Return the dimension of `x`.

dim(x) <- value Set the dimension of `x` to `value`.

dimnames(x) Return the dimnames of `x`.

getExternalStorage(x) Return the `storage` slot of `x`.

getNativeStorageMethods Return `getNativeStorageMethods(x@storage)`.

is.finite(x) Return an external vector of logicals with same length as `x` indicating if an element of `x` is finite.

is.infinite(x) Return an external vector of logicals with same length as `x` indicating if an element of `x` is infinite.

is.na(x) Return an external vector of logicals with same length as `x` indicating if an element of `x` is `NA`.

is.nan(x) Return an external vector of logicals with same length as `x` indicating if an element of `x` is `NaN`.

length(x) Return the length of `x`.

length(x) <- value Set the length of `x` to `value`.

names(x) Return the names of `x`.

names(x) <- value Set the names of `x` to `value`.

lapply(X, FUN, ...) Lapply for "externalVectorWithStorage" class.

setDimNames(x, value) Set the dimnames of `x` to `value` without any error checking or coercion. Return the modified `x`.

setNames(x, value) Set the names of `x` to `value` without any error checking or coercion. Return the modified `x`.

rebind(x, length.x = length(x), type = defaultElement(x), names.x = names(x), dim.x = dim(x), dimnames.x = dim Create an `"externalVectorWithStorage"` object with the same external storage backend class as `x` but with (possibly) new length, type, names, dim or dimnames. If arrayOnly is FALSE, the names, dim and dimnames for the new object are not set.

initialize(.Object, ...) Initialize a new object from class `"externalVectorWithSTorage"`.

See Also

`externalVector` and `externalMatrix` for how how to create new objects from this class easily.

`gcAllocator-class` *Class "gcAllocator", memory allocator class for external resources*

Description

A memory allocator class for external resources that allocates memory using R memory management system. The allocated memory is freed by the R garbage collector when all refernce to it is removed.

Objects from the Class

Objects can be created by calls of the form `new("gcAllocator")`.

Extends

Class "externalAllocator", directly.

Methods

Signature components for the methods:

resource	The class "externalResource"
alloc	The class of "gcAllocator"
size	The class "ANY"
type	The class "ANY"
copy	The class "logical"
value	The class "ANY"

Description of the methods:

allocate(resource, alloc, size, type, ...): Allocate the external pointer in `resource`. If `type` is a basic vector object, then allocate an object of same mode with length `size`. Otherwise allocate `size` bytes of raw memory. The allocation is done by creating an R basic vector with same class as type and storing it in the protected field of the newly created external pointer. Ends with a call to `initializeResource` to initialize the `resource` object.

deallocate(resource, alloc): Replace the protected field of `resource@ptr` with `R_NilValue`.

external.size Return the size of the allocated memory in `resource`.

external.size<-(resource, copy, alloc, value): If `value` is same as `external.size(resource)`, then no action is taken. Otherwise, reallocate the memory in `resource` with new size `value`. If `copy` is `TRUE` (the default), then the new memory is initialized to the content of the old memory for the minimum of old and new sizes. Content of any uninitialized memory is undefined.

reinitializePointer(resource, alloc): If the object `resource` was saved as an R image (by serialization code, by saving the R workspace, or by an explicit call to `save`) then the raw memory pointer in any `"externalptr"` object in it would be set to `0`. This method reinitializes the memory (if possible) by using the protected field of the external pointer.

Examples

```
library(externalVector)
## set a storage class
setClass("testStorage",
         representation(ptr="externalptr"),
         contains="externalResource")
setMethod("initializeResource", "testStorage",
          function(resource, ptr, size, type, ...)
       {
           resource@ptr <- ptr
           resource
       })
setMethod("allocator", "testStorage",
          function(resource)
          new("gcAllocator"))
setMethod("getPointer", "testStorage",
          function(resource)
          resource@ptr)
setMethod("allocatedSize", "testStorage",
```

```
          function(resource)
          32)
## Now create an object from the class
x <- new("testStorage")
x
external.size(x)
external.size(x) <- 64
x
external.size(x)
deallocate(x)
x
external.size(x)
```

`getExternalStorage` *Extract external storage*

Description

This generic extracts the "externalStorage" object associated with an "externalVector" object.

Usage

```
getExternalStorage(x)
```

Arguments

`x`	An "externalVector" object.

Value

An "externalStorage" object.

See Also

`externalStorage-class`, `externalVectorWithStorage-class`

`getNativeStorageMethods`
Return the storage methods associated with an object

Description

For an object from a subclass of "externalStorage" or for an object with an associated subclass of "externalStorage", this generic returns the `"externalptr"` stored in the `"nativeStorageMethods"` object associated with that subclass of "externalStorage".

Usage

```
getNativeStorageMethods(x)
```

Arguments

x	An object.

Value

An "externalptr" object with a pointer to a C storageMethods structure.

See Also

nativeStorageMethods-class

---

indirectExternalMatrix-class

Class "indirectExternalMatrix", class for indirect matrix subset of an externalVector object.

Description

Objects from this class store an object of class "externalVector" and a matrix subscript for that object and behaves as if this is an object created by applying that subscript.

Objects from the Class

Objects can be created by calls of the form as("indirectExternalMatrix", vec) where vec is an object of class "externalVector".

Slots

actual: Object of class "externalVector", the actual object of which this is a subset

index: Object of class "matrixIndex", the vector subscript to get the subset.

Extends

Class "externalVector", directly.

Methods

e1 op e2 For any arithmetic or comparison operation op this returns the result of the operation. Either (or both) of e1 and e2 are of class "indirectExternalMatrix".

Complex(z) Return the result of an operation in the Complex group generic.

Math(x) Return the result of an operation in the Math group generic.

Summary(x) Return the result of an operation in the Summary group generic.

log(x) Return natural logarithm of x.

log(x, base) Return logarithm of x with base base.

x[i, j, . . . , drop=FALSE] Extract a subset of x according to the given subscripts.

x[i, j, . . . <- value] Subassign of x by value according to the given subscripts. This assigns to the correct elements of x@actual.

as.Rvector(x) Coerce x to a basic R vector with same names, dimension and dimnames (if any of these are present in x).

as.vector(x, mode) Return an R basic vector with mode `mode`.

as.matrix Return a matrix of appropriate mode .

length(x) Return the length of `x`.

length(x) <- value Set the length of `x` to `value`.

dim(x) Return the dimension of `x`.

dim(x) <- value Set the dimension of `x` to `value`.

dimnames(x) Return the dimnames of `x`.

dimnames(x) <- value Set the dimnames of `x` to `value`.

getExternalStorage(x) Return the `storage` slot of associated with `x@actual`.

is.finite(x) Return a vector of logicals with same length as `x` indicating if an element of `x` is finite.

is.infinite(x) Return a vector of logicals with same length as `x` indicating if an element of `x` is infinite.

is.na(x) Return a vector of logicals with same length as `x` indicating if an element of `x` is `NA`.

is.nan(x) Return a vector of logicals with same length as `x` indicating if an element of `x` is `NaN`.

force return the result of `force(x@actual)[x@index]`.

See Also

`indirectExternalVector-class` for the matrix class corresponding to this.

`indirectExternalVector-class`
Class "indirectExternalVector", class for indirect vector subset of an externalVector object.

Description

Objects from this class store an object of class `"externalVector"` and a vector subscript for that object and behaves as if this is an object created by applying that subscript.

Objects from the Class

Objects can be created by calls of the form `as("indirectExternalVector", vec)` where `vec` is an object of class `"externalVector"`.

Slots

`actual`**:** Object of class `"externalVector"`, the actual object of which this is a subset

`index`**:** Object of class `"vectorIndex"`, the vector subscript to get the subset.

Extends

Class `"externalVector"`, directly.

Methods

e1 op e2 For any arithmetic or comparison operation op this returns the result of the operation. Either (or both) of e1 and e2 are of class "indirectExternalVector".

Complex(z) Return the result of an operation in the Complex group generic.

Math(x) Return the result of an operation in the Math group generic.

Summary(x) Return the result of an operation in the Summary group generic.

log(x) Return natural logarithm of x.

log(x, base) Return logarithm of x with base base.

x[i, j, . . . , drop=FALSE] Extract a subset of x according to the given subscripts.

x[i, j, . . . <- value] Subassign of x by value according to the given subscripts. This assigns to the correct elements of x@actual.

as.Rvector(x) Coerce x to a basic R vector with same names, dimension and dimnames (if any of these are present in x).

as.vector(x, mode) Return an R basic vector with mode mode.

as.matrix Return a matrix of appropriate mode .

length(x) Return the length of x.

length(x) <- value Set the length of x to value.

dim(x) Return the dimension of x.

dim(x) <- value Set the dimension of x to value.

dimnames(x) Return the dimnames of x.

dimnames(x) <- value Set the dimnames of x to value.

getExternalStorage(x) Return the storage slot of associated with x@actual.

is.finite(x) Return a vector of logicals with same length as x indicating if an element of x is finite.

is.infinite(x) Return a vector of logicals with same length as x indicating if an element of x is infinite.

is.na(x) Return a vector of logicals with same length as x indicating if an element of x is NA.

is.nan(x) Return a vector of logicals with same length as x indicating if an element of x is NaN.

setNames(x, value) Set the names of x to value without any error checking or coercion. Return the modified x.

force return the result of force(x@actual)[x@index].

See Also

indirectExternalMatrix-class for the matrix class corresponding to this.

internalType *Get a vector object similar to the one stored inside an external storage object*

Description

Get a vector object similar to the one stored inside an external storage object.

Usage

```
internalType(x)
```

Arguments

x	An "externalStorage" object.

Value

An R basic vector object representing the kind of vector object stored in the x.

See Also

externalStorage-class

makeSlice *Create a new object of class "sliceIndex"*

Description

Create a new object of class "sliceIndex"

Usage

```
makeSlice(start = 1, length.slice = 1, stride = 1)
```

Arguments

start	Start of the index.
length.slice	Length of the index.
stride	Difference between consecutive elements of the index.

Value

An object of class "sliceIndex".

See Also

sliceIndex-class

Examples

```
x <- makeSlice(0, 4, 2)
as(x, "positiveIndex")
```

`matrixIndex-class` *Class "matrixIndex", index class representing a set of row and column subscripts for a matrix*

Description

Objects of class `"matrixIndex"` represent subscripts corresponding to a submatrix of some matrix of a given size. Like the `"vectorIndex"` subclasses, this is a class for internal use.

Objects from the Class

Objects can be created by calls of the form `new("matrixIndex", row, col, origdim, Names)`.

Slots

`row`**:** Object of class `"vectorIndex"`, the row subscript.

`col`**:** Object of class `"vectorIndex"`, the column subscript.

`origdim`**:** Object of class `"integer"`, the dimension of the full matrix.

`Names`**:** Object of class `"vectorNamesType"`, names for the submatrix elements when the submatrix is treated as vectors.

Extends

Class `"externalVector"`, directly.

Methods

Signature components for the methods are:

x	The class "allIndex"
scalar	Length one positive "integer"
i	The class "ANY"
j	The class "ANY"
drop	The class "logical"
value	The class "ANY"
.Object	The class "allIndex"

x[i, j, drop=FALSE] Create another object of cloass `matrixIndex` with same `origdims` represents a further submatrix of the full matrix. If `drop` is `TRUE` and at least one of the dimensions of the answer is `1`, the answer is an object of class `"vectorIndex"` instead.

x[[i]] Get the subscript representing the indices for dimension `i`.

dim(x) Get the dimension of `x`.

dimnames(x) Get the dimnames of `x`.

length(x) Get the length of `x` considered as a vector. This is same as `prod(dim(x))`.

names(x) Get the names of x.

setDimNames(x, value) Set the dimnames of x and return the modified x.

setNames(x, value) Set the names of x and return the modified x.

as(x, "vectorIndex") Coerce x to an object of class "vectorIndex".

See Also

vectorIndex-class for the class of row and column subscripts.

NAIndex-class *Class "NAIndex", index class representing a subscript of all NA's for a vector or an array dimension*

Description

When used as a subscript for a vector object x, this is equivalent to rep(NA, length(x)). When used as a subscript for the i-th dimension of an array object x, this is equivalent to rep(NA, dim(x)[i]).

Objects from the Class

Objects can be created by calls of the form new("NAIndex", Length, Names). Length must be a non-negative integer. Default value for Length is 0. The Names argument, if not missing, must be of length Length. If missing, Names is taken to be NULL.

Slots

.Data: Object of class "logical", this is always same as NA.

Length: Object of class "integer", this is the length of the index.

Names: Object of class "vectorNamesType", this contains names associated with the elements of the index (if any).

Extends

Class "logical", directly. Class "vectorIndex", directly.

Methods

Signature components for the methods are:

x	The class "allIndex"
scalar	Length one positive "integer"
i	The class "ANY"
value	The class "ANY"
.Object	The class "allIndex"

x+scalar: Add scalar to x. Result is x.

scalar+x: Add scalar to x. Result is x.

x*scalar: Multiply x by scalar. Result is x.

scalar*x: Multiply x by scalar. Result is x.

dim(x) <- value Set a dimension on x. If value is of length 2, this may create a "matrixIndex".

x[i]: Subset x. If length of i is 0, the result is an object of class "noneIndex". Otherwise, the result is an object of class "NAIndex" with same length as i.

x[]: Subset x with the subscript missing. Result is x.

x[i] <- value: Subassign x. Results in an error (as objects from class "allIndex" are not mutable.

x[[i]] <- value: Subassign x. Results in an error (as objects from class "allIndex" are not mutable.

allNA(x): Are all elements of x NA? Result is always FALSE.

anyNA(x): Is any element of x NA? Result is always FALSE.

initialize(.Object, Length=0, Names): Initialize an object from class "allIndex".

length(x): Return the length of x which is always Inf.

length(x) <- value: Modify the length of x. If value is 0, makes x an object of class "noneIndex". Otherwise, modifies length of x to value.

names(x): Returns the Names slot of x.

names(x) <- value: Sets the Names slot of x to value.

See Also

vectorIndex-class for the super class of "NAIndex".

---

nativeStorageMethods-class

Class "nativeStorageMethods", class holding an external pointer to a C storeageMethods structure.

Description

This simple class has only one purpose - holding an external pointer that contains a pointer to a C storageMethods structure. The memory for the C structure is allocated by the "gcAllocator" memory allocator.

Objects from the Class

Objects can be created by calls of the form new("nativeStorageMethods", ...). For details on how the ... argument to new works, see documentation for the "initialize" method for class "externalResource".

The ptr slot in a newly created object contains pointer to an uninitialized C storageMethods structure.

Slots

ptr: Object of class "externalptr"

Extends

Class "externalResource", directly.

Methods

Signature components of the methods:

resource	The class "nativeStorageMethods"
ptr	The class "externalptr"
size	The class "ANY"
type	The class "ANY"

Description of the methods:

allocatedSize(resource): Returns the size of the C `storageMethods` structure in bytes.

allocator(resource): Returns an object of class `"gcAllocator"`.

getPointer(reource): Return the `ptr` slot.

initializeResource(resource, ptr, size, type, ...): Sets the `ptr`

See Also

`nativeStorageMethodsList-class` for how this class is used.

`externalResource-class` for more on the super class of this class.

`nativeStorageMethodsList-class`
Class "nativeStorageMethodsList", contains objects from class "nativeStorageMethods"

Description

The class `"nativeStorageMethodsList"` contains a number of slots, all from class `"nativeStorageMethods"`. Usually, each object of class `"nativeStorageMethodsList"` is associated with a particular subclass of `"externalSTorage"`.

Objects from this class are never visible to the user.

Objects from the Class

Objects can be created by calls of the form `new("nativeStorageMethodsList")`. Any extra arguments are ignored.

Slots

All slots in this class are objects of class `"nativeStorageMethods"`.

`allMethods`**:** Contains the C `storageMethods` structure with all methods registered (directly or indirectly) for the associated `"externalStorage"` class.

`logicalMethods`**:** Object returned by `getExternalStorageMethods` for the associated `"externalStorage"` class and type `"logical"`.

`integerMethods`**:** Object returned by `getExternalStorageMethods` for the associated `"externalStorage"` class and type `"integer"`.

`numericMethods`**:** Object returned by `getExternalStorageMethods` for the associated `"externalStorage"` class and type `"numeric"`.

`complexMethods`**:** Object returned by `getExternalStorageMethods` for the associated `"externalStorage"` class and type `"complex"`.

`characterMethods`: Object returned by `getExternalStorageMethods` for the associated `"externalStorage"` class and type `"character"`.

`listMethods`: Object returned by `getExternalStorageMethods` for the associated `"externalStorage"` class and type `"list"`.

Methods

initialize(.Object, ...) Initializer method for class `"nativeStorageMethodsList"`. This initializes each of the slots of the class. Any extra argument in `...` is ignored.

`noneIndex-class` *Class "noneIndex", index class representing a subscript for none of the elements of a vector or an array dimension*

Description

When used as a subscript for a vector object or for any dimension of an array object this is equivalent to `integer(0)`.

Objects from the Class

Objects can be created by calls of the form `new("noneIndex")`. Objects from this class can not be modified and they are all equivalent.

Slots

`.Data`: Object of class `"integer"`, this is always equal to `integer(0)`.

Extends

Class `"integer"`, directly. Class `"vectorIndex"`, directly.

Methods

Signature components for the methods are:

x	The class "allIndex"
scalar	Length one positive "integer"
i	The class "ANY"
value	The class "ANY"
.Object	The class "allIndex"

`x+scalar`: Add `scalar` to `x`. Result is `x`.

`scalar+x`: Add `scalar` to `x`. Result is `x`.

`x*scalar`: Multiply `x` by `scalar`. Result is `x`.

`scalar*x`: Multiply `x` by `scalar`. Result is `x`.

dim(x) <- value Set a dimension on `x`. If value is of length 2, this may create a "matrixIndex".

`x[i]`: Subset x. The result is an object of class `"NAIndex"` of length same as `length(integer(0)[i])`.

`x[]`: Subset x with the subscript missing. Result is x.

`x[i] <- value`: Subassign x. Results in an error (as objects from class "noneIndex" are not mutable.

`x[[i]] <- value`: Subassign x. Results in an error (as objects from class "noneIndex" are not mutable.

`allNA(x)`: Are all elements of x NA? Result is always FALSE.

`anyNA(x)`: Is any element of x NA? Result is always FALSE.

`initialize(.Object)`: Initialize an object from class "allIndex".

`length(x)`: Return the length of x which is always 0.

`length(x) <- value`: Modify the length of x. Always results in an error.

See Also

vectorIndex-class for the super class of "noneIndex".

---

nonSlotAttributes *Functions to extract and set attributes which are not slots.*

Description

These functions access an object's attribute list. The first form returns the an object's attribute list after removing any attribute which is also a slot for the object. The assignment form makes the list on the right-hand side of the assignment the object's attribute list (if appropriate).

Usage

```
nonSlotAttributes(x)
nonSlotAttributes(x) <- value
```

Arguments

x	An object.
value	An appropriate attribute list, or NULL.

See Also

attributes and attributes<-

`positiveIndex-class` *Class "positiveIndex", index class representing a sequence of non-negative integers*

Description

This represents a subscript of arbitrary non-negative integers.

Objects from the Class

Objects can be created by calls of the form `new("positiveIndex", val)`. If `val` is missing, the result is an object of class `"positiveIndex"` of length `0`. Otherwise, it is coerced to an integer and an object of class `"positiveIndex"` is created which has the same `.Data` slot as `val`. The `rangeIndex` and `noNA` slots are calculated by the initialization code.

Slots

`.Data`**:** Object of class `"integer"`, the actual index values.

`rangeIndex`**:** Object of class `"integer"` of length 2, same as `range(.Data)`.

`noNA`**:** Object of class `"logical"` of length 1, same as `!any(is.na(.Data))`.

Extends

Class `"integer"`, directly. Class `"vectorIndex"`, directly.

Methods

Signature components for the methods are:

x	The class "allIndex"
scalar	Length one positive "integer"
i	The class "ANY"
value	The class "ANY"
.Object	The class "allIndex"

`x+scalar` Add `scalar` to `x`. Result is `x` with `x@.Data` replaced by `x@.Data+scalar`.

`scalar+x` Add `scalar` to `x`. Result is `x` with `x@.Data` replaced by `x@.Data+scalar`.

`x*scalar` Multiply `x` by `scalar`. Result is `x` with `x@.Data` replaced by `x@.Data*scalar`.

`scalar*x` Multiply `x` by `scalar`. Result is `x` with `x@.Data` replaced by `x@.Data*scalar`.

dim(x) <- value Set a dimension on `x`. If value is of length 2, this may create a "matrixIndex".

`x[i]` Subset `x`. If length of `i` is `0`, the result is an object of class `"noneIndex"`. Otherwise, the result is an object of class `"sliceIndex"` or of class `"positiveIndex"`.

`x[]` Subset x with the subscript missing. Result is x.

`allNA(x)` Are all elements of `x` `NA`? Result is `all(is.na(x@.Data))`.

`anyNA(x)` Is any element of `x` `NA`? Result is `any(is.na(x@.Data))`.

`as.integer(x)` Convert `x` to an R subscript - adds 1 to each element of `x@.Data`.

`initialize(.Object, .Data)` Initialize `.Object`.

rebind *Allocate a new external vector with external storage*

Description

This function allocates a new "externalVectorWithStorage" object with possibly new length, type, names, dimension or dimnames.

Usage

```
rebind(x, length.x = length(x), type = x@storage@type,
       names.x = x@Names, dim.x = x@Dim,
       dimnames.x = x@DimNames, ..., arrayOnly = FALSE)
```

Arguments

x	An object of class "externalVectorWithStorage".
length.x	New length.
type	New type.
names.x	New names.
dim.x	New dimension.
dimnames.x	New dimnames.
...	Other arguments passed to initializer of the class of x.
arrayOnly	Just create the data part with no names, dimensions or dimnames.

Value

A new object of same class as x.

See Also

externalVectorWithStorage-class

setDimNames *Set names and dimnames*

Description

These generics set the names and the dimnames of an object without any sanity checks.

Usage

```
setNames(object, nm)
setDimNames(object, dnm)
```

Arguments

object	An object.
nm	The new names.
dnm	The new dimnames.

Value

The object after modification.

See Also

names<-, dimnames<-

---

setExternalStorageClass

Create subclass of externalStorage

Description

Function to create a subclass of "externalStorage" and associate native C methods with the class.

Usage

```
setExternalStorageClass(Class, methodNames, ..., contains = "externalStorage", where = topenv(pa
```

Arguments

Class	Character string, name for the class.
methodNames	Character vector, names of the native C functions for use as methods (see Adding Native Methods below).
...	Other arguments passed to setClass.
contains	what classes does this class extend? (These are called superclasses in some languages.) Should have at least one class name which is a subclass of "externalStorage".
where	The environment in which to store or remove the definition. Defaults to the top-level environment of the calling function (the global environment for ordinary computations, but the environment or namespace of a package when loading that package).

Details

This function does two things. It creates a subclass of "externalStorage". It also associates a set of given C methods as native methods for this subclass of "externalStorage".

Adding Native Methods

To make the external vectors as efficient as possible, the interface between the storage method classes and the "externalVectorWithSTorage" class uses a set of C function pointers for each subclass of "externalStorage". There are two types of methods - those which are for a particular type of vector and those which are not for a particular type. For a type xxx where xxx is one of logical, integer, numeric, complex or character, the type specific functions are

xxxGetElt	get an element from vector of type xxx
xxxSetElt	set an element in a vector of type xxx
xxxSubset	subset a vector of type xxx
xxxSubassign	subassign a vector of type xxx
xxxGetMatrixElt	get an element from matrix of type xxx
xxxSetMatrixElt	set an element in a matrix of type xxx
xxxMatrixSubset	subset a matrix of type xxx
xxxMatrixSubassign	subassign a matrix of type xxx

The functions not specific to any type are

alloc	allocate a new storage object
size	return the length of the vector in the storage object
resize	modify the length of the vector in the storage object

The subset and subassign functions for vectors and matrices have reasonable default implementations in terms of the element get/set functions - so they need not be implemented for each "externalStorage" subclass.

The `methodNames` argument to `setExternalStorageClass` must be used to register any C function for use as a particular method. This argument must be a character vector with elements of the form `methodName=functionName` where `methodName` is one of the method names given in the tables above and function name is a C function name which has been registered as a C function with R using the foreign function registration mechanism. See the chapter on "System and foreign language interfaces" in "Writing R Extensions" in the 'doc/manual' subdirectory of the R source tree for more details on registering C functions.

See Also

`setClass` for possible arguments in .... `externalStorage-class` for the "externalStorage" class.

Examples

```
## Not run:
setExternalStorageClass("simpleStorage", c(logicalGetElt="simpleLogicalGetElt",
                                            logicalSetElt="simpleLogicalSetElt",
                                            numericGetElt="simpleNumericGetElt",
                                            numericSetElt="simpleNumericSetElt",
                                            size="simpleSize",
                                            resize="simpleResize"
                                            alloc="simpleAlloc"),
                        contains = "externalStorage")

## End(Not run)
```

`setVirtualMethod` *Set a virtual method for a given signature*

Description

This sets a method for a signature. If there is no generic with the given name, it also creates the appropriate generic. If called, the method generates an error indicating the classes of the arguments which are part of the signature.

Usage

```
setVirtualMethod(name, signature.virtual, signature.nonvirtual = character(), ..., where = topen
```

Arguments

name	The character-string name of the generic function.
signature.virtual	Signature of the arguments for which the virtual method is being set. All of these arguments must have a virtual class in the signature.
signature.nonvirtual	Signature for other arguments. Any ommitted argument is assumed to have class "ANY".
...	Other arguments. See details.
where	The database in which to store the definition of the method.

Details

If the generic for name is not defined, an attempt is made to create it with `setGeneric(name, ..., where = where)`.

Value

This method exists for its side-effect of setting up the virtual method.

The return value is the result from the call to setMethod to create the virtual method.

See Also

setGeneric, setMethod

Examples

```
setClass("myMatrix")
setVirtualMethod("dim", "myMatrix")
setVirtualMethod("dimnames", "myMatrix")
setClass("myMatrix2", representation(val="numeric",
                                     d = "integer",
                                     dn = "list"),
         contains="myMatrix")
x <- new("myMatrix2", val=1:4, d = as.integer(c(2, 2)),
         dn = list(LETTERS[1:2], letters[1:2]))
## A call dim(x) or dimnames(x) would generate an error here
setMethod("dim", "myMatrix", function(x) x@d)
setMethod("dimnames", "myMatrix", function(x) x@dn)
dim(x)
dimnames(x)
```

`simpleStorage-class` *Class "simpleStorage", an external storage class that uses reference to R basic vectors as storage*

Description

The class `"simpleStorage"` is an implementation of the `"externalStorage"` class. The actual storage for the vector object is contained in an R basic vector referenced through the `ptr` slot.

Objects from the Class

Objects can be created by calls of the form `new("simpleStorage", type, length)`. See the `initialize` method for `"externalStorage"` for details.

Slots

`ptr`: Object of class `"externalptr"`, keeps the R basic vector in the protected field. For `"logical"`, `"integer"`, `"numeric"`, `"complex"`, the address field of the external pointer also holds a pointer to the data in the basic R vector.

`type`: Object of class `"vector"`, a vector object of length one representing the type of object stored in the external storage. Usually one of the basic R vector types.

`length`: Object of class `"integer"`, Cached value for the length of the vector stored in the external storage.

Extends

Class `"externalStorage"`, directly. Class `"externalResource"`, by class `"externalStorage"`.

Methods

Signature components for the methods:

resource The class "simpleStorage"

Description of the methods:

allocatedSize(resource) Length of the vector to be stored.

allocatedType The type of vector to be stored.

See Also

`externalStorage-class` to see details of the super class.

`externalVector` to create `"externalVector"` objects that use this resource.

`sliceIndex-class` *Class "sliceIndex", index class representing a sequence of non-negative integers*

Description

This represents a sequence that can be created with `seq(start, by=stride, length=n)` where `star >= 0` and `start+(n-1)*stride >= 0`.

Objects from the Class

Objects can be created by calls of the form `makeSlice(start, length, stride)`.

Slots

`content`: Object of class `"numeric"` of length `3`. The first element of `content` is the start of the index, the second element is the length of the index, and the third element is the difference between successive elements of the index.

`Names`: Object of class `"vectorNamesType"`, this contains names associated with the elements of the index (if any).

Extends

Class `"vectorIndex"`, directly.

Methods

Signature components for the methods are:

x	The class "allIndex"
scalar	Length one positive "integer"
i	The class "ANY"
value	The class "ANY"
.Object	The class "allIndex"

`x+scalar`: Add `scalar` to `x`. Result is `x` with `x@content[1]` replaced by `x@content[1]+scalar`.

`scalar+x`: Add `scalar` to `x`. Result is `x` with `x@content[1]` replaced by `x@content[1]+scalar`.

`x*scalar`: Multiply `x` by `scalar`. Result is `x` with `x@content[3]` replaced by `x@content[3]*scalar`.

`scalar*x`: Multiply `x` by `scalar`. Result is `x` with `x@content[3]` replaced by `x@content[3]*scalar`.

`x[i]`: Subset `x`. If length of `i` is `0`, the result is an object of class `"noneIndex"`. Otherwise, the result is an object of class `"sliceIndex"` or of class `"positiveIndex"`.

dim(x) <- value Set a dimension on `x`. If value is of length 2, this may create a "matrixIndex".

`x[]`: Subset x with the subscript missing. Result is x.

`allNA(x)`: Are all elements of `x` `NA`? Result is always `FALSE`.

`anyNA(x)`: Is any element of `x` `NA`? Result is always `FALSE`.

`length(x)`: Return the length of `x`.

`length(x) <- value`: Modify the length of `x`. If `value` is `0`, makes `x` an object of class `"noneIndex"`. Otherwise, modifies length of `x` to `value`.

`names(x)`: Returns the `Names` slot of `x`.

`names(x) <- value` : Sets the `Names` slot of `x` to `value`.

`SubscriptList` *Function to convert a list of array subscripts to a list objects of class vectorIndex.*

Description

SubscriptList takes a list of subscripts to a subset or subassign operation and converts each subscript to an object belonging to the virtual class `vectorIndex`.

Usage

```
SubscriptList(subs, length.object, dim.object = NULL, names.object = NULL, dimnames.object = NUL
```

Arguments

`subs`	An object of class "list", elements are valid subscripts.
`length.object`	A scalar of class "integer", representing the length of the object being subsetted or subassigned.
`dim.object`	An object of class "integer" or `NULL`, the dimension of the object.
`names.object`	An object belonging to class union "validNamesClass", the names of the object viewed as a vector.
`dimnames.object`	`NULL` or an object of class "list", each of whose elements are objects belonging to class union "validNamesClass".
`assign.operation`	A scalar of class "logical", if `TRUE` the subscripts are for a subasign operation, if `FALSE`, the subscripts are for a subset operation.

Value

An object of class "list" of same length as `subs` with each element a

Author(s)

Saikat DebRoy <<saikat@stat.wisc.edu>>

See Also

See class union `vectorNamesType-class` for possible subscripts which are matched against the `names.object` or elements of `dimnames.object`.

Also see class union `vectorIndex-class` for some other possible subscripts and for all possible elements in the returned list.

Finally, see `[` for other subscripts that are valid.

Examples

```
subs <- alist(i= , j= )
subs$j <- 1:4
SubscriptList(subs, 15, c(3, 5))
SubscriptList(list(i=c("a", "c"), j=-2), 15, c(3, 5),
              dimnames=list(letters[1:3], LETTERS[1:5]))
```

Subset *Generics Subset and Subset2 in package 'externalVector' and their methods*

Description

The generics Subset and Subset2 and their methods are used internally and are not for general use. They may be replaced by C code in future.

---

union-class *Some useful class unions*

Description

The class "listOrNull" is a union of the classes "list" and "NULL". The class "listOrNull" is a union of the classes "list" and "NULL". The class "vectorNamesType" is a union of the classes "character", "NULL" and "externalVectorWithStorage".

Objects from the Class

These are virtual classes: no objects may be created from them.

Methods

No methods defined with class "listOrNull", "integerOrNull" or "vectorNamesType" in the signature.

---

var *Correlation, Variance and Covariance (Matrices)*

Description

`var`, `cov` and `cor` compute the variance of `x` and the covariance or correlation of `x` and `y` if these are vectors. If `x` and `y` are matrices then the covariances (or correlations) between the columns of `x` and the columns of `y` are computed.

`cov2cor` scales a covariance matrix into the corresponding correlation matrix *efficiently*.

Usage

```
var(x, y, na.rm = FALSE, use)

cov(x, y, use = "all.obs",
    method = c("pearson", "kendall", "spearman"))

cor(x, y, use = "all.obs",
     method = c("pearson", "kendall", "spearman"))

cov2cor(V)
```

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a vector, matrix or data frame with compatible dimensions to x. The default is equivalent to y = x (but more efficient).
na.rm	logical. Should missing values be removed?
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "all.obs", "complete.obs" or "pairwise.complete.obs".
method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
V	symmetric numeric matrix, usually positive definite such as a covariance matrix.

Details

For cov and cor one must *either* give a matrix or data frame for x *or* give both x and y.

var is just another interface to cov, where na.rm is used to determine the default for use when that is unspecified. If na.rm is TRUE then the complete observations (rows) are used (use = "complete") to compute the variance. Otherwise (use = "all"), var will give an error if there are missing values.

If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by casewise deletion. Finally, if use has the value "pairwise.complete.obs" then the correlation between each pair of variables is computed using all complete pairs of observations on those variables. This can result in covariance or correlation matrices which are not positive semidefinite.

The denominator $n - 1$ is used which gives an unbiased estimator of the (co)variance for i.i.d. observations. These functions return NA when there is only one observation (whereas S-PLUS has been returning NaN), and fail if x has length zero.

For cor(), if method is "kendall" or "spearman", Kendall's τ or Spearman's ρ statistic is used to estimate a rank-based measure of association. These are more robust and have be recommended if the data do not necessarily come from a bivariate normal distribution.
For cov(), a non-Pearson method is unusual but available for the sake of completeness. Note that "spearman" basically computes cor(R(x), R(y)) (or cov(.,.)) where R(u) := rank(u, na.last="keep").

Scaling a covariance matrix into a correlation one can be achieved in many ways, mathematically most appealing by multiplication with a diagonal matrix from left and right, or more efficiently by using sweep(.., FUN = "/") twice. The cov2cor function is even a bit more efficient, and provided mostly for didactical reasons.

Value

For `r <- cor(*, use = "all.obs")`, it is now guaranteed that `all(r <= 1)`.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth \& Brooks/Cole.

See Also

`cor.test` (package **ctest**) for confidence intervals (and tests).

`cov.wt` for *weighted* covariance computation.

`sd` for standard deviation (vectors).

Examples

```
x <- externalNumeric(10)
x[] <- 1:10
var(x)# 9.166667

var(x[1:5],x[1:5])# 2.5

## Two simple vectors
cor(x,x+1)# == 1
```

---

vectorIndex-class *Class "vectorIndex", base class for index classes used to represent vector subscripts*

Description

Together, all subclasses of this class represent the possible representations of vector subscripts and subscripts for individual array dimensions.

Usually users would not create objects from any of the subclasses of `"vectorIndex"` directly.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "vectorIndex" in the signature.

See Also

`allIndex-class`, `noneIndex-class`, `NAIndex-class`, `sliceIndex-class`, `positiveIndex-class`, all of which are subclasses of`"vectorIndex"`.

Index

∗Topic **array**
externalMatrix, 9
SubscriptList, 38
var, 39
∗Topic **attribute**
nonSlotAttributes, 30
setDimNames, 32
∗Topic **classes**
allIndex-class, 2
as.Rvector, 5
asEach, 6
exprMatrix-class, 7
externalAllocator-class, 8
externalResource-class, 10
externalStorage-class, 12
externalVector, 13
externalVector-class, 14
externalVectorWithStorage-class, 16
gcAllocator-class, 17
getExternalStorage, 19
getNativeStorageMethods, 19
indirectExternalMatrix-class, 20
indirectExternalVector-class, 21
internalType, 23
makeSlice, 23
matrixIndex-class, 24
NAIndex-class, 25
nativeStorageMethods-class, 26
nativeStorageMethodsList-class, 28
noneIndex-class, 29
positiveIndex-class, 31
rebind, 32
setExternalStorageClass, 33
simpleStorage-class, 36
sliceIndex-class, 37
union-class, 39
vectorIndex-class, 41
∗Topic **list**
SubscriptList, 38
∗Topic **methods**
allIndex-class, 2
allNA, 3
allocators, 4
defaultElement, 7
externalAllocator-class, 8
externalResource-class, 10
nativeStorageMethods-class, 26
setVirtualMethod, 34
Subset, 39
∗Topic **multivariate**
var, 39
∗Topic **programming**
getNativeStorageMethods, 19
setExternalStorageClass, 33
∗Topic **univar**
var, 39
[, *38*
[,NAIndex,ANY,ANY,ANY-method *(*NAIndex-class*)*, 25
[,NAIndex,allIndex,ANY,ANY-method *(*NAIndex-class*)*, 25
[,NAIndex,noneIndex,ANY,ANY-method *(*NAIndex-class*)*, 25
[,allIndex,ANY,missing,missing-method *(*allIndex-class*)*, 2
[,allIndex,missing,missing,missing-method *(*allIndex-class*)*, 2
[,externalVectorWithStorage,ANY,ANY,ANY-method *(*externalVectorWithStorage-class*)*, 16
[,indirectExternalMatrix,ANY,ANY,ANY-method *(*indirectExternalMatrix-class*)*, 20
[,indirectExternalMatrix,ANY,ANY,missing-method *(*indirectExternalMatrix-class*)*, 20
[,indirectExternalMatrix,ANY,missing,logical-method *(*indirectExternalMatrix-class*)*, 20
[,indirectExternalMatrix,ANY,missing,missing-method *(*indirectExternalMatrix-class*)*, 20
[,indirectExternalMatrix,missing,ANY,logical-method *(*indirectExternalMatrix-class*)*, 20

[,indirectExternalMatrix,missing,ANY,missing-method (indirectExternalMatrix-class), 20
[,indirectExternalMatrix,missing,missing,logical-method (indirectExternalMatrix-class), 20
[,indirectExternalMatrix,missing,missing,missing-method (indirectExternalMatrix-class), 20
[,indirectExternalVector,ANY,missing,ANY-method (indirectExternalVector-class), 21
[,indirectExternalVector,missing,missing,ANY-method (indirectExternalVector-class), 21
[,matrixIndex,ANY,ANY,ANY-method (matrixIndex-class), 24
[,noneIndex,ANY,missing,missing-method (noneIndex-class), 29
[,noneIndex,logical,missing,missing-method (noneIndex-class), 29
[,noneIndex,missing,missing,missing-method (noneIndex-class), 29
[,positiveIndex,ANY,missing,ANY-method (positiveIndex-class), 31
[,positiveIndex,positiveIndex,missing,ANY-method (positiveIndex-class), 31
[,positiveIndex,sliceIndex,missing,ANY-method (positiveIndex-class), 31
[,sliceIndex,ANY,missing,ANY-method (sliceIndex-class), 37
[,sliceIndex,allIndex,missing,ANY-method (sliceIndex-class), 37
[,sliceIndex,noneIndex,missing,ANY-method (sliceIndex-class), 37
[,sliceIndex,positiveIndex,missing,ANY-method (sliceIndex-class), 37
[,sliceIndex,sliceIndex,missing,ANY-method (sliceIndex-class), 37
[<-,NAIndex,ANY,ANY,ANY-method (NAIndex-class), 25
[<-,allIndex,ANY,ANY,ANY-method (allIndex-class), 2
[<-,externalVectorWithStorage,ANY,ANY,ANY-method (externalVectorWithStorage-class), 16
[<-,externalVectorWithStorage,ANY,ANY,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
[<-,indirectExternalMatrix,ANY,ANY,ANY-method (indirectExternalMatrix-class), 20
[<-,indirectExternalMatrix,missing,missing,ANY-method (indirectExternalMatrix-class), 20
[<-,indirectExternalVector,ANY,missing,ANY-method (indirectExternalVector-class), 21
[<-,indirectExternalVector,missing,missing,ANY-method (indirectExternalVector-class), 21
[<-,noneIndex,ANY,ANY,ANY-method (noneIndex-class), 29
[[,matrixIndex,numeric,missing-method (matrixIndex-class), 24
[[<-,NAIndex-method (NAIndex-class), 25
[[<-,allIndex-method (allIndex-class), 2
[[<-,noneIndex-method (noneIndex-class), 29

allIndex-class, 2
allNA, 3
allNA,allIndex-method (allIndex-class), 2
allNA,ANY-method (allNA), 3
allNA,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
allNA,indirectExternalVector-method (indirectExternalVector-class), 21
allNA,matrixIndex-method (matrixIndex-class), 24
allNA,NAIndex-method (NAIndex-class), 25
allNA,noneIndex-method (noneIndex-class), 29
allNA,positiveIndex-method (positiveIndex-class), 31
allNA,sliceIndex-method (sliceIndex-class), 37
allocate (allocators), 4
allocate,externalResource,externalAllocator-method (externalAllocator-class), 8
allocate,externalResource,gcAllocator-method (gcAllocator-class), 17
allocate,externalResource,missing-method (externalResource-class), 10
allocatedSize (allocators), 4
allocatedSize,externalResource-method (externalResource-class), 10
allocatedSize,nativeStorageMethods-method (nativeStorageMethods-class), 26
allocatedSize,simpleStorage-method (simpleStorage-class), 36

allocatedType *(allocators)*, 4
allocatedType,externalResource-method *(externalResource-class)*, 10
allocatedType,simpleStorage-method *(simpleStorage-class)*, 36
allocator *(allocators)*, 4
allocator,externalResource-method *(externalResource-class)*, 10
allocator,externalStorage-method *(externalStorage-class)*, 12
allocator,nativeStorageMethods-method *(nativeStorageMethods-class)*, 26
allocators, 4
anyNA *(allNA)*, 3
anyNA,allIndex-method *(allIndex-class)*, 2
anyNA,ANY-method *(allNA)*, 3
anyNA,indirectExternalMatrix-method *(indirectExternalMatrix-class)*, 20
anyNA,indirectExternalVector-method *(indirectExternalVector-class)*, 21
anyNA,matrixIndex-method *(matrixIndex-class)*, 24
anyNA,NAIndex-method *(NAIndex-class)*, 25
anyNA,noneIndex-method *(noneIndex-class)*, 29
anyNA,positiveIndex-method *(positiveIndex-class)*, 31
anyNA,sliceIndex-method *(sliceIndex-class)*, 37
apply,externalVector-method *(externalVector-class)*, 14
Arith,allIndex,numeric-method *(allIndex-class)*, 2
Arith,ANY,externalVectorWithStorage-method *(externalVectorWithStorage-class)*, 16
Arith,ANY,indirectExternalMatrix-method *(indirectExternalMatrix-class)*, 20
Arith,ANY,indirectExternalVector-method *(indirectExternalVector-class)*, 21
Arith,ANY,vectorIndex-method *(vectorIndex-class)*, 41
Arith,externalVectorWithStorage,ANY-method *(externalVectorWithStorage-class)*, 16
Arith,externalVectorWithStorage,externalVectorWithStorage-method *(externalVectorWithStorage-class)*, 16
Arith,externalVectorWithStorage,missing-method *(externalVectorWithStorage-class)*, 16
Arith,indirectExternalMatrix,ANY-method *(indirectExternalMatrix-class)*, 20
Arith,indirectExternalMatrix,indirectExternalMatrix-met *(indirectExternalMatrix-class)*, 20
Arith,indirectExternalMatrix,indirectExternalVector-met *(indirectExternalMatrix-class)*, 20
Arith,indirectExternalVector,ANY-method *(indirectExternalVector-class)*, 21
Arith,indirectExternalVector,indirectExternalMatrix-met *(indirectExternalMatrix-class)*, 20
Arith,indirectExternalVector,indirectExternalVector-met *(indirectExternalVector-class)*, 21
Arith,NAIndex,numeric-method *(NAIndex-class)*, 25
Arith,noneIndex,numeric-method *(noneIndex-class)*, 29
Arith,numeric,allIndex-method *(allIndex-class)*, 2
Arith,numeric,NAIndex-method *(NAIndex-class)*, 25
Arith,numeric,noneIndex-method *(noneIndex-class)*, 29
Arith,numeric,positiveIndex-method *(positiveIndex-class)*, 31
Arith,numeric,sliceIndex-method *(sliceIndex-class)*, 37
Arith,positiveIndex,numeric-method *(positiveIndex-class)*, 31
Arith,sliceIndex,numeric-method *(sliceIndex-class)*, 37
Arith,vectorIndex,ANY-method *(vectorIndex-class)*, 41
as, *6*
as.character,externalVector-method *(externalVector-class)*, 14
as.complex,externalVector-method *(externalVector-class)*, 14
as.data.frame,externalVector-method *(externalVector-class)*, 14
as.double,externalVector-method *(externalVector-class)*, 14

as.integer,externalVector-method
(externalVector-class), 14
as.integer,positiveIndex-method
(positiveIndex-class), 31
as.list,externalVector-method
(externalVector-class), 14
as.logical,externalVector-method
(externalVector-class), 14
as.matrix, *6*
as.matrix,externalVector-method
(externalVector-class), 14
as.matrix,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20
as.matrix,indirectExternalVector-method
(indirectExternalVector-class),
21
as.numeric,externalVector-method
(externalVector-class), 14
as.Rmatrix (as.Rvector), 5
as.Rmatrix,ANY-method (as.Rvector), 5
as.Rmatrix,matrix-method (as.Rvector), 5
as.Rvector, 5
as.Rvector,array-method (as.Rvector), 5
as.Rvector,character-method
(as.Rvector), 5
as.Rvector,complex-method (as.Rvector),
5
as.Rvector,externalVector-method
(externalVector-class), 14
as.Rvector,externalVectorWithStorage-method
(externalVectorWithStorage-class),
16
as.Rvector,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20
as.Rvector,indirectExternalVector-method
(indirectExternalVector-class),
21
as.Rvector,integer-method (as.Rvector),
5
as.Rvector,list-method (as.Rvector), 5
as.Rvector,logical-method (as.Rvector),
5
as.Rvector,matrix-method (as.Rvector), 5
as.Rvector,numeric-method (as.Rvector),
5
as.single,externalVector-method
(externalVector-class), 14
as.vector, *6*
as.vector,externalVector-method
(externalVector-class), 14
as.vector,indirectExternalMatrix,ANY-method
(indirectExternalMatrix-class),
20
as.vector,indirectExternalMatrix,missing-method
(indirectExternalMatrix-class),
20
as.vector,indirectExternalVector,ANY-method
(indirectExternalVector-class),
21
as.vector,indirectExternalVector,missing-method
(indirectExternalVector-class),
21
asEach, 6
asEach,ANY,character-method (asEach), 6
asEach,externalVector,character-method
(externalVector-class), 14
asEach,externalVectorWithStorage,character-method
(externalVectorWithStorage-class),
16
attributes, *30*

c,externalVector-method
(externalVector-class), 14
character, *14*
coerce,externalVector,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20
coerce,externalVector,indirectExternalVector-method
(indirectExternalVector-class),
21
coerce,externalVector,matrix-method
(externalVector-class), 14
coerce,externalVector,vector-method
(externalVector-class), 14
coerce,indirectExternalMatrix,matrix-method
(indirectExternalMatrix-class),
20
coerce,indirectExternalMatrix,vector-method
(indirectExternalMatrix-class),
20
coerce,indirectExternalVector,matrix-method
(indirectExternalVector-class),
21
coerce,indirectExternalVector,vector-method
(indirectExternalVector-class),
21
coerce,integer,positiveIndex-method
(positiveIndex-class), 31
coerce,matrix,externalVector-method
(externalVector-class), 14
coerce,matrix,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20

coerce,matrixIndex,vectorIndex-method (matrixIndex-class), 24
coerce,numeric,positiveIndex-method (positiveIndex-class), 31
coerce,sliceIndex,positiveIndex-method (sliceIndex-class), 37
coerce,vector,externalVector-method (externalVector-class), 14
coerce,vector,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
coerce,vector,indirectExternalVector-method (indirectExternalVector-class), 21
colMeans,externalVectorWithStorage,ANY,missing-method (externalVectorWithStorage-class), 16
colMeans,externalVectorWithStorage,logical,numeric-method (externalVectorWithStorage-class), 16
colMeans,externalVectorWithStorage,missing,ANY-method (externalVectorWithStorage-class), 16
colMeans,externalVectorWithStorage,missing,missing-method (externalVectorWithStorage-class), 16
colSums,externalVectorWithStorage,ANY,missing-method (externalVectorWithStorage-class), 16
colSums,externalVectorWithStorage,logical,numeric-method (externalVectorWithStorage-class), 16
colSums,externalVectorWithStorage,missing,ANY-method (externalVectorWithStorage-class), 16
colSums,externalVectorWithStorage,missing,missing-method (externalVectorWithStorage-class), 16
Compare,ANY,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
Compare,ANY,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
Compare,ANY,indirectExternalVector-method (indirectExternalVector-class), 21
Compare,externalVectorWithStorage,ANY-method (externalVectorWithStorage-class), 16
Compare,externalVectorWithStorage,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
Compare,indirectExternalMatrix,ANY-method (indirectExternalMatrix-class), 20
Compare,indirectExternalMatrix,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
Compare,indirectExternalMatrix,indirectExternalVector-method (indirectExternalMatrix-class), 20
Compare,indirectExternalVector,ANY-method (indirectExternalVector-class), 21
Compare,indirectExternalVector,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
Compare,indirectExternalVector,indirectExternalVector-method (indirectExternalVector-class), 21
complex, *14*
Complex,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
Complex,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
Complex,indirectExternalVector-method (indirectExternalVector-class), 21
cor (var), 39
cor,ANY,ANY-method (var), 39
cor,ANY,externalVector-method (externalVector-class), 14
cor,ANY,missing-method (var), 39
cor,externalVector,ANY-method (externalVector-class), 14
cor.test, *41*
cov (var), 39
cov,ANY,ANY-method (var), 39
cov,ANY,externalVector-method (externalVector-class), 14
cov,ANY,missing-method (var), 39
cov,externalVector,ANY-method (externalVector-class), 14
cov.wt, *41*
cov2cor (var), 39
cov2cor,externalVector-method (externalVector-class), 14

deallocate (allocators), 4
deallocate,externalResource,externalAllocator-method (externalAllocator-class), 8

deallocate,externalResource,gcAllocator-method (gcAllocator-class), 17
deallocate,externalResource,missing-method (externalResource-class), 10
defaultElement, 7
defaultElement,externalVector-method (externalVector-class), 14
defaultElement,vector-method (defaultElement), 7
dim,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
dim,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
dim,indirectExternalVector-method (indirectExternalVector-class), 21
dim,matrixIndex-method (matrixIndex-class), 24
dim<-,allIndex,ANY-method (allIndex-class), 2
dim<-,externalVectorWithStorage,ANY-method (externalVectorWithStorage-class), 16
dim<-,externalVectorWithStorage,NULL-method (externalVectorWithStorage-class), 16
dim<-,indirectExternalMatrix,ANY-method (indirectExternalMatrix-class), 20
dim<-,indirectExternalMatrix,NULL-method (indirectExternalMatrix-class), 20
dim<-,indirectExternalVector,ANY-method (indirectExternalVector-class), 21
dim<-,NAIndex,ANY-method (NAIndex-class), 25
dim<-,noneIndex,ANY-method (noneIndex-class), 29
dim<-,positiveIndex,ANY-method (positiveIndex-class), 31
dim<-,sliceIndex,ANY-method (sliceIndex-class), 37
dimnames,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
dimnames,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
dimnames,indirectExternalVector-method (indirectExternalVector-class), 21
dimnames,matrixIndex-method (matrixIndex-class), 24
dimnames<-,externalVector,list-method (externalVector-class), 14
dimnames<-,externalVector,NULL-method (externalVector-class), 14
dimnames<-,indirectExternalMatrix,ANY-method (indirectExternalMatrix-class), 20
dimnames<-,indirectExternalVector,ANY-method (indirectExternalVector-class), 21

exprMatrix-class, 7
external.size (allocators), 4
external.size,externalResource,externalAllocator-method (externalAllocator-class), 8
external.size,externalResource,gcAllocator-method (gcAllocator-class), 17
external.size,externalResource,missing-method (externalResource-class), 10
external.size<- (allocators), 4
external.size<-,externalResource,ANY,missing,ANY-method (externalResource-class), 10
external.size<-,externalResource,logical,externalAlloca (externalAllocator-class), 8
external.size<-,externalResource,logical,gcAllocator,nu (gcAllocator-class), 17
externalAllocator-class, 8
externalCharacter (externalVector), 13
externalComplex (externalVector), 13
externalInteger (externalVector), 13
externalList (externalVector), 13
externalLogical (externalVector), 13
externalMatrix, 9, *17*
externalNumeric (externalVector), 13
externalResource-class, 10
externalStorage-class, 12
externalVector, 13, *17*, *36*
externalVector-class, 14
externalVectorWithStorage-class, 16

force,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
force,indirectExternalVector-method (indirectExternalVector-class), 21

gcAllocator-class, 17
getExternalStorage, 19

getExternalStorage,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
getExternalStorage,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
getExternalStorage,indirectExternalVector-method (indirectExternalVector-class), 21
getNativeStorageMethods, 19
getNativeStorageMethods,ANY-method (getNativeStorageMethods), 19
getNativeStorageMethods,externalStorage-method (externalStorage-class), 12
getNativeStorageMethods,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
getPointer (allocators), 4
getPointer,externalResource-method (externalResource-class), 10
getPointer,externalStorage-method (externalStorage-class), 12
getPointer,nativeStorageMethods-method (nativeStorageMethods-class), 26

indirectExternalMatrix-class, 20
indirectExternalVector-class, 21
initialize,allIndex-method (allIndex-class), 2
initialize,externalResource-method (externalResource-class), 10
initialize,externalStorage-method (externalStorage-class), 12
initialize,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
initialize,NAIndex-method (NAIndex-class), 25
initialize,nativeStorageMethodsList-method (nativeStorageMethodsList-class), 28
initialize,noneIndex-method (noneIndex-class), 29
initialize,positiveIndex-method (positiveIndex-class), 31
initializeResource (allocators), 4
initializeResource,externalResource,externalptr-method (externalResource-class), 10
initializeResource,externalStorage-method (externalStorage-class), 12
initializeResource,nativeStorageMethods-method (nativeStorageMethods-class), 26
integer, *14*
integerOrNull-class (union-class), 39
internalType, 23
internalType,externalStorage-method (externalStorage-class), 12
is.finite,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
is.finite,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
is.finite,indirectExternalVector-method (indirectExternalVector-class), 21
is.infinite,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
is.infinite,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
is.infinite,indirectExternalVector-method (indirectExternalVector-class), 21
is.na,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
is.na,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
is.na,indirectExternalVector-method (indirectExternalVector-class), 21
is.nan,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
is.nan,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
is.nan,indirectExternalVector-method (indirectExternalVector-class), 21

lapply,externalVector-method (externalVector-class), 14
lapply,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
length,allIndex-method (allIndex-class), 2
length,externalVectorWithStorage-method (externalVectorWithStorage-class), 16

length,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
length,indirectExternalVector-method (indirectExternalVector-class), 21
length,matrixIndex-method (matrixIndex-class), 24
length,NAIndex-method (NAIndex-class), 25
length,noneIndex-method (noneIndex-class), 29
length,sliceIndex-method (sliceIndex-class), 37
length<-,allIndex-method (allIndex-class), 2
length<-,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
length<-,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
length<-,indirectExternalVector-method (indirectExternalVector-class), 21
length<-,NAIndex-method (NAIndex-class), 25
length<-,noneIndex-method (noneIndex-class), 29
length<-,sliceIndex-method (sliceIndex-class), 37
list, *14*
listOrNull-class (union-class), 39
log,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
log,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
log,indirectExternalVector-method (indirectExternalVector-class), 21
logical, *14*

makeSlice, 23
match,ANY,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
match,externalVectorWithStorage,ANY-method (externalVectorWithStorage-class), 16
Math,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
Math,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
Math,indirectExternalVector-method (indirectExternalVector-class), 21
matrix, *9*
matrixIndex-class, 24
mean,externalVector-method (externalVector-class), 14
median,externalVector,ANY-method (externalVector-class), 14
median,externalVector,missing-method (externalVector-class), 14

NA, *40*
NAIndex-class, 25
names,externalVectorWithStorage-method (externalVectorWithStorage-class), 16
names,indirectExternalMatrix-method (indirectExternalMatrix-class), 20
names,indirectExternalVector-method (indirectExternalVector-class), 21
names,matrixIndex-method (matrixIndex-class), 24
names,NAIndex-method (NAIndex-class), 25
names,sliceIndex-method (sliceIndex-class), 37
names<-,externalVector,ANY-method (externalVector-class), 14
names<-,externalVector,externalVector-method (externalVector-class), 14
names<-,externalVector,NULL-method (externalVector-class), 14
names<-,externalVectorWithStorage,NULL-method (externalVectorWithStorage-class), 16
names<-,indirectExternalMatrix,ANY-method (indirectExternalMatrix-class), 20
names<-,indirectExternalVector,NULL-method (indirectExternalVector-class), 21
names<-,NAIndex,ANY-method (NAIndex-class), 25
names<-,NAIndex,externalVector-method (NAIndex-class), 25
names<-,sliceIndex,ANY-method (sliceIndex-class), 37

names<-,sliceIndex,externalVector-method
(sliceIndex-class), 37
nativeStorageMethods-class, 26
nativeStorageMethodsList-class, 28
noneIndex-class, 29
nonSlotAttributes, 30
nonSlotAttributes<-
(nonSlotAttributes), 30
numeric, *14*

Ops,ANY,externalVectorWithStorage-method
(externalVectorWithStorage-class),
16
Ops,ANY,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20
Ops,ANY,indirectExternalVector-method
(indirectExternalVector-class),
21
Ops,externalVectorWithStorage,ANY-method
(externalVectorWithStorage-class),
16
Ops,indirectExternalMatrix,ANY-method
(indirectExternalMatrix-class),
20
Ops,indirectExternalMatrix,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20
Ops,indirectExternalMatrix,indirectExternalVector-method
(indirectExternalMatrix-class),
20
Ops,indirectExternalVector,ANY-method
(indirectExternalVector-class),
21
Ops,indirectExternalVector,indirectExternalMatrix-method
(indirectExternalMatrix-class),
20
Ops,indirectExternalVector,indirectExternalVector-method
(indirectExternalVector-class),
21

positiveIndex-class, 31

rebind, 32
rebind,externalVectorWithStorage-method
(externalVectorWithStorage-class),
16
reinitializePointer *(allocators)*, 4
reinitializePointer,externalResource,externalAllocator-method
(externalAllocator-class), 8
reinitializePointer,externalResource,gcAllocator-method
(gcAllocator-class), 17
reinitializePointer,externalResource,missing-method
(externalResource-class), 10
rowMeans,externalVectorWithStorage,ANY,missing-method
(externalVectorWithStorage-class),
16
rowMeans,externalVectorWithStorage,logical,numeric-meth
(externalVectorWithStorage-class),
16
rowMeans,externalVectorWithStorage,missing,ANY-method
(externalVectorWithStorage-class),
16
rowMeans,externalVectorWithStorage,missing,missing-meth
(externalVectorWithStorage-class),
16
rowSums,externalVectorWithStorage,ANY,missing-method
(externalVectorWithStorage-class),
16
rowSums,externalVectorWithStorage,logical,numeric-metho
(externalVectorWithStorage-class),
16
rowSums,externalVectorWithStorage,missing,ANY-method
(externalVectorWithStorage-class),
16
rowSums,externalVectorWithStorage,missing,missing-metho
(externalVectorWithStorage-class),
16

sapply,externalVector-method
(externalVector-class), 14
sd, *41*
setClass, *34*
setDimNames, 32
setDimNames,externalVector-method
(externalVector-class), 14
setDimNames,externalVectorWithStorage-method
(externalVectorWithStorage-class),
16
setDimNames,matrixIndex-method
(matrixIndex-class), 24
setExternalStorageClass, *13*, 33
setGeneric, *35*
setMethod, *35*
setNames *(setDimNames)*, 32
setNames,externalVector-method
(externalVector-class), 14
setNames,externalVectorWithStorage-method
(externalVectorWithStorage-class),
16
setNames,indirectExternalVector-method
(indirectExternalVector-class),
21
setNames,matrixIndex-method
(matrixIndex-class), 24

setVirtualMethod, *9*, *12*, 34
show,externalVector-method *(externalVector-class)*, 14
simpleStorage-class, 36
sliceIndex-class, 37
SubscriptList, 38
Subset, 39
Subset,ANY,allIndex-method *(Subset)*, 39
Subset,ANY,ANY-method *(Subset)*, 39
Subset,ANY,matrixIndex-method *(Subset)*, 39
Subset,ANY,NAIndex-method *(Subset)*, 39
Subset,ANY,noneIndex-method *(Subset)*, 39
Subset,ANY,positiveIndex-method *(Subset)*, 39
Subset,ANY,sliceIndex-method *(Subset)*, 39
Subset-methods *(Subset)*, 39
Subset2 *(Subset)*, 39
Subset2,ANY,allIndex,allIndex-method *(Subset)*, 39
Subset2,ANY,allIndex,ANY-method *(Subset)*, 39
Subset2,ANY,allIndex,NAIndex-method *(Subset)*, 39
Subset2,ANY,ANY,allIndex-method *(Subset)*, 39
Subset2,ANY,ANY,ANY-method *(Subset)*, 39
Subset2,ANY,ANY,NAIndex-method *(Subset)*, 39
Subset2,ANY,ANY,noneIndex-method *(Subset)*, 39
Subset2,ANY,ANY,positiveIndex-method *(Subset)*, 39
Subset2,ANY,ANY,sliceIndex-method *(Subset)*, 39
Subset2,ANY,NAIndex,allIndex-method *(Subset)*, 39
Subset2,ANY,NAIndex,ANY-method *(Subset)*, 39
Subset2,ANY,NAIndex,NAIndex-method *(Subset)*, 39
Subset2,ANY,noneIndex,ANY-method *(Subset)*, 39
Subset2,ANY,noneIndex,noneIndex-method *(Subset)*, 39
Subset2,ANY,positiveIndex,ANY-method *(Subset)*, 39
Subset2,ANY,positiveIndex,positiveIndex-method *(Subset)*, 39
Subset2,ANY,sliceIndex,ANY-method *(Subset)*, 39
Subset2,ANY,sliceIndex,sliceIndex-method *(Subset)*, 39
Subset2-methods *(Subset)*, 39
Summary,externalVectorWithStorage,logical-method *(externalVectorWithStorage-class)*, 16
Summary,externalVectorWithStorage,missing-method *(externalVectorWithStorage-class)*, 16
Summary,externalVectorWithStorage-method *(externalVectorWithStorage-class)*, 16
Summary,indirectExternalMatrix,ANY-method *(indirectExternalMatrix-class)*, 20
Summary,indirectExternalMatrix-method *(indirectExternalMatrix-class)*, 20
Summary,indirectExternalVector,ANY-method *(indirectExternalVector-class)*, 21
Summary,indirectExternalVector-method *(indirectExternalVector-class)*, 21
sweep, *40*

union-class, 39

var, 39
var,ANY,ANY-method *(var)*, 39
var,ANY,externalVector-method *(externalVector-class)*, 14
var,ANY,missing-method *(var)*, 39
var,externalVector,ANY-method *(externalVector-class)*, 14
vector, *14*
vectorIndex-class, 41
vectorNamesType-class *(union-class)*, 39