

Package ‘apex’

July 22, 2025

Title Phylogenetic Methods for Multiple Gene Data

Version 1.0.6

Description Toolkit for the analysis of multiple gene data (Jombart et al. 2017) <doi:10.1111/1755-0998.12567>.

'apex' implements the new S4 classes 'multidna', 'multiplyDat' and associated methods to handle aligned DNA sequences from multiple genes.

Depends R (>= 3.1.3), methods, ape, phangorn

Imports utils, graphics, stats, adegenet

License GPL (>= 2)

URL <https://github.com/thibautjombart/apex>

BugReports <https://github.com/thibautjombart/apex/issues>

Collate doc.R internal.R multidna.class.R multiplyDat.class.R
add.gaps.R rm.gaps.R show.R multidna.constructor.R
multiplyDat.constructor.R accessors.R subset.R concatenate.R
plot.R readfiles.R datasets.R dist.R getTree.R exports.R

VignetteBuilder knitr

Suggests testthat, knitr, rmarkdown

Encoding UTF-8

RoxygenNote 7.3.0

NeedsCompilation no

Author Klaus Schliep [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-2941-0161>>),
Thibaut Jombart [aut],
Zhian Namir Kamvar [aut],
Eric Archer [aut],
Rebecca Harris [aut]

Maintainer Klaus Schliep <klaus.schliep@gmail.com>

Repository CRAN

Date/Publication 2024-01-31 19:40:07 UTC

Contents

accessors	2
add.gaps	4
concatenate	5
dist.multidna	6
getTree	7
initialize,multidna-method	8
initialize,multiplyDat-method	10
multidna-class	11
multidna2alignment	12
multidna2genind	13
multidna2multiplyDat	14
multiplyDat-class	15
plot,multidna-method	16
read.multidna	17
rm.gaps	18
show,multidna-method	19
show,multiplyDat-method	19
[,multidna,ANY,ANY,ANY-method	20
Index	21

accessors	<i>multidna Accessors</i>
-----------	---------------------------

Description

Accessors for slots in [multidna](#) and [multiplyDat](#) objects.

Usage

```
getNumInd(x, ...)  
  
## S4 method for signature 'multidna'  
getNumInd(x, ...)  
  
## S4 method for signature 'multiplyDat'  
getNumInd(x, ...)  
  
getNumLoci(x, ...)  
  
## S4 method for signature 'multidna'  
getNumLoci(x, ...)  
  
## S4 method for signature 'multiplyDat'  
getNumLoci(x, ...)
```

```
getLocusNames(x, ...)  
  
## S4 method for signature 'multidna'  
getLocusNames(x, ...)  
  
## S4 method for signature 'multiplyDat'  
getLocusNames(x, ...)  
  
setLocusNames(x) <- value  
  
## S4 replacement method for signature 'multidna'  
setLocusNames(x) <- value  
  
## S4 replacement method for signature 'multiplyDat'  
setLocusNames(x) <- value  
  
getNumSequences(x, ...)  
  
## S4 method for signature 'multidna'  
getNumSequences(x, exclude.gap.only = TRUE,  
  loci = NULL, ...)  
  
## S4 method for signature 'multiplyDat'  
getNumSequences(x, exclude.gap.only = TRUE,  
  loci = NULL, ...)  
  
getSequenceNames(x, ...)  
  
## S4 method for signature 'multidna'  
getSequenceNames(x, exclude.gap.only = TRUE,  
  loci = NULL, ...)  
  
## S4 method for signature 'multiplyDat'  
getSequenceNames(x, exclude.gap.only = TRUE,  
  loci = NULL, ...)  
  
getSequences(x, ...)  
  
## S4 method for signature 'multidna'  
getSequences(x, loci = NULL, ids = NULL,  
  simplify = TRUE, exclude.gap.only = TRUE, ...)  
  
## S4 method for signature 'multiplyDat'  
getSequences(x, loci = NULL, ids = NULL,  
  simplify = TRUE, exclude.gap.only = TRUE, ...)
```

Arguments

x a [multidna](#) or [multiplyDat](#) object.

...	further arguments passed on to other functions.
value	a replacement value for the slot.
exclude.gap.only	logical. Remove or ignore sequences containing all gaps?
loci	a character, numeric, or logical vector identifying which loci to return.
ids	a character, numeric, or logical vector identifying which sequences to return within a locus.
simplify	logical. If FALSE, always return a list of DNABin sequences. If TRUE and only one locus has been requested, return a single DNABin object.

Details

getNumInd Returns the number of individuals.
getNumLoci Returns the number of loci.
getLocusNames Returns the names of each locus.
setLocusNames Sets the names of each locus.
getNumSequences Returns the number of sequences in each locus.
getSequenceNames Returns the names of individual sequences at each locus.
getSequences Returns sequences of specified loci and individuals.

Value

returns the information stored in a slot, see details.

add.gaps	<i>Add gap-only sequences for missing data</i>
----------	--

Description

In [multidna](#) and [multiplyDat](#), some individuals may not be sequenced for all genes. The generic function `add.gaps` has method for both objects; it identifies the missing sequences, and adds gap-only sequences to the alignments wherever needed.

Usage

```
add.gaps(x, ...)

## S4 method for signature 'multidna'
add.gaps(x, ...)

## S4 method for signature 'multiplyDat'
add.gaps(x, ...)
```

Arguments

x	a multidna or multiplyDat object.
...	further arguments passed to other methods (currently not used).

concatenate*Concatenate genes into a single matrix*

Description

These functions concatenate separate DNA alignments into a single alignment matrix. `concatenate` is a generic with methods for:

- `multidna`: returns a DNAbin matrix
- `multiplyDat`: returns a `phyDat` object

Usage

```
concatenate(x, ...)  
  
## S4 method for signature 'multidna'  
concatenate(x, genes = TRUE, ...)  
  
## S4 method for signature 'multiplyDat'  
concatenate(x, genes = TRUE, ...)
```

Arguments

<code>x</code>	a multidna or a multiplyDat object.
<code>...</code>	further arguments passed to other methods (currently not used).
<code>genes</code>	an optional vector indicating the genes to retain for the concatenation; any way to subset the list in <code>x@dna</code> is acceptable; by default, all genes are used.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```
## simple conversion with nicely ordered output  
data(woodmouse)  
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])  
x <- new("multidna", genes)  
x  
plot(x)  
  
image(concatenate(x))
```

dist.multidna	<i>Pairwise distances for multiple gene data</i>
---------------	--

Description

This function computes pairwise genetic distances between individuals using genes in a [multidna](#) object. By default, one distance matrix (dist object) is created for each each, but a single distance can be derived by pooling all genes (argument pool=TRUE)

Usage

```
dist.multidna(x, pool = FALSE, genes = TRUE, ...)
```

Arguments

x	a multidna object.
pool	a logical indicating if all genes should be pooled (concatenated) to obtain a single distance matrix; defaults to FALSE.
genes	an optional vector indicating the genes to retain for the concatenation; any way to subset the list in x@dna is acceptable; by default, all genes are used.
...	further arguments passed to dist.dna .

Value

a list of dist objects (pool=FALSE) or a single dist object (pool=TRUE)

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

See Also

[dist.dna](#)

Examples

```
## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
plot(x)

## get separate distance matrix and pooled one
1D <- dist.multidna(x)
D <- dist.multidna(x, pool=TRUE)

## get corresponding NJ trees
```

```
ltrees <- lapply(lD, nj)
tree <- nj(D)

opar <- par(no.readonly=TRUE)
par(mfrow=c(3,1))
for(i in 1:2) plot(ltrees[[i]], main=names(ltrees)[i])
plot(tree, main="Pooled distances")
par(opar)
```

getTree

Build phylogenies from multiple gene data

Description

This function builds separate phylogenetic trees for each gene in a [multidna](#) object, specifying a method for computing pairwise distances between individuals, and a method to build the tree from the distance matrix. By default, procedures from ape are used.

Usage

```
getTree(x, pool = FALSE, genes = TRUE, model = "N",
        pairwise.deletion = TRUE, method = nj, ladderize = TRUE,
        negative.branch.length = FALSE, ...)
```

Arguments

x	a multidna object.
pool	a logical indicating if all genes should be pooled (concatenated) to obtain a single tree; defaults to FALSE.
genes	an optional vector indicating the genes to retain for the concatenation; any way to subset the list in x@dna is acceptable; by default, all genes are used.
model	a character string passed to dist.dna describing the model to be used to compute genetic distances; defaults to 'N', the absolute number of mutations separating sequences.
pairwise.deletion	a logical passed to dist.dna indicating if pairwise deletions should be used; the alternative is to remove all sites for which at least one missing value is present.
method	a function building a tree from a matrix of pairwise genetic distances.
ladderize	a logical indicating if the tree should be ladderized; defaults to TRUE.
negative.branch.length	a logical indicating if negative branch lengths should be allowed (e.g. in the case of Neighbor-Joining reconstruction), or not, in which case they are set to 0 (FALSE, default).
...	further arguments passed to the tree reconstruction method defined by 'method'.

Value

a multiPhylo object

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

See Also

[dist.multidna](#)

Examples

```
## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
plot(x)

## make trees, default parameters
trees <- getTree(x)
trees
plot(trees, type="unrooted")

## make one single tree based on concatenated genes
tree <- getTree(x, pool=TRUE)
tree
plot(tree, type="unrooted")
```

initialize,multidna-method

multidna constructor

Description

New [multidna](#) objects can be created using `new("multidna", ...)` where `"..."` are arguments documented below. The main input is a list of DNABin matrices. The constructor ensures that all matrices will be reordered in the same way, and as an option (setting `add.gaps=TRUE`, gap-only sequences (`"...—..."`) will be added wherever sequences are missing.

Usage

```
## S4 method for signature 'multidna'
initialize(.Object, dna = NULL, ind.info = NULL,
  gene.info = NULL, add.gaps = TRUE, quiet = FALSE, ...)
```


Arguments

<code>.Object</code>	the object skeleton, automatically generated when calling <code>new</code> .
<code>dna</code>	a list of DNABin matrices (1 per gene); rows should be labelled and indicate individuals, but different individuals and different orders can be used in different matrices.
<code>ind.info</code>	an optional data.frame containing information on the individuals, where individuals are in rows.
<code>gene.info</code>	an optional data.frame containing information on the genes, where genes are in rows.
<code>add.gaps</code>	a logical indicating if gap-only sequences should be used where sequences are missing; defaults to TRUE.
<code>quiet</code>	a logical indicating if messages should be shown; defaults to FALSE.
<code>...</code>	further arguments to be passed to other methods

Value

an object of class `multidna` containing alignments.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

See Also

- the `multidna` class
- `read.multidna` and `read.multidna`

Examples

```
## empty object
new("multidna")

## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
image(woodmouse)
image(x@dna[[1]])
image(x@dna[[2]])

## trickier conversion with missing sequences / wrong order
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[c(5:1,14:15),501:965])
x <- new("multidna", genes)
x
image(x@dna[[1]])
image(x@dna[[2]])
```

```
initialize,multiplyDat-method
      multiplyDat constructor
```

Description

New [multiplyDat](#) objects can be created using `new("multiplyDat", ...)` where `"..."` are arguments documented below. The main input is a list of `phyDat` matrices. The constructor ensures that all matrices will be reordered in the same way, and genes with missing individuals will be filled by sequences of gaps ("-").

Usage

```
## S4 method for signature 'multiplyDat'
initialize(.Object, seq = NULL,
  type = character(0), ind.info = NULL, gene.info = NULL,
  add.gaps = TRUE, quiet = FALSE, ...)
```

Arguments

<code>.Object</code>	the object skeleton, automatically generated when calling <code>new</code> .
<code>seq</code>	a list of <code>phyDat</code> matrices (1 per gene); rows should be labelled and indicate individuals, but different individuals and different orders can be used in different matrices.
<code>type</code>	a character string indicating the type of the sequences stored: "DNA" for DNA sequences, "AA" for amino-acids.
<code>ind.info</code>	an optional <code>data.frame</code> containing information on the individuals, where individuals are in rows.
<code>gene.info</code>	an optional <code>data.frame</code> containing information on the genes, where genes are in rows.
<code>add.gaps</code>	a logical indicating if gap-only sequences should be used where sequences are missing; defaults to <code>TRUE</code> .
<code>quiet</code>	a logical indicating if messages should be shown; defaults to <code>FALSE</code> .
<code>...</code>	further arguments to be passed to other methods

Value

an object of class [multiplyDat](#) containing alignments.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>
 Thibaut Jombart <t.jombart@imperial.ac.uk>

See Also

- the [multiplyDat](#) class
- [read.multiplyDat](#)

Examples

```
data(Laurasiatherian)
#' ## empty object
new("multiplyDat")

## simple conversion with nicely ordered output
genes <- list(gene1=Laurasiatherian[, 1:1600],
              gene2=Laurasiatherian[, 1601:3179])
x <- new("multiplyDat", genes)
x

## trickier conversion with missing sequences / wrong order
genes <- list(gene1=Laurasiatherian[1:40,],
              gene2=Laurasiatherian[8:47, ])
x <- new("multiplyDat", genes)
x
```

multidna-class

multidna: class for multiple gene data

Description

This formal (S4) class is used to store multiple DNA alignments. Sequences are stored as a (possibly named) list, with each element of the list being a separate DNA alignment stored as a DNABin matrix. The rows of the separate matrices all correspond to the same individuals, ordered identically.

Slots

`dna` a list of DNABin matrices; empty slot should be NULL

`labels` a vector of labels of individuals

`n.ind` the number of individuals

`n.seq` the total number of sequences (pooling all genes), including gap sequences

`n.seq.miss` the total number of gap-only sequences

`ind.info` a data.frame containing information on the individuals, where individuals are in rows; empty slot should be NULL

`gene.info` a data.frame containing information on the genes, where genes are in rows; empty slot should be NULL

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```
## empty object
new("multidna")

## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
image(woodmouse)
image(x@dna[[1]])
image(x@dna[[2]])

## trickier conversion with missing sequences / wrong order
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[c(5:1,14:15),501:965])
x <- new("multidna", genes)
x
image(x@dna[[1]])
image(x@dna[[2]])
```

multidna2alignment	<i>Convert from multidna into alignment (seqinr)</i>
--------------------	--

Description

The functions `multidna2alignment` and `multiplyDat2alignment` concatenates separate sequences and return an alignment object of the `seqinr` package.

Usage

```
multidna2alignment(x, genes = TRUE)

multiplyDat2alignment(x, genes = TRUE)
```

Arguments

<code>x</code>	a multidna or multiplyDat object.
<code>genes</code>	an optional vector indicating the genes to retain for the concatenation; any way to subset the list in <code>x@dna</code> or <code>x@seq</code> is acceptable; by default, all genes are used.

Value

a alignment object

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>, Zhian N. Kamvar, Klaus Schliep

See Also

- concatenate
- [as.alignment](#) to convert single DNABin objects.

Examples

```
## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
y <- multidna2alignment(x)
y
x2 <- multidna2multiplyDat(x)
z <- multiplyDat2alignment(x2)
```

multidna2genind	<i>Convert multidna into genind</i>
-----------------	-------------------------------------

Description

The functions `multidna2genind` and `multiplyDat2genind` concatenates separate DNA alignments, and then extracts SNPs of the resulting alignment into a [genind](#) object.

Usage

```
multidna2genind(x, genes = TRUE, mlst = FALSE, gapIsNA = FALSE)
```

```
multiplyDat2genind(x, genes = TRUE, mlst = FALSE, gapIsNA = FALSE)
```

Arguments

<code>x</code>	a multidna or multiplyDat object.
<code>genes</code>	an optional vector indicating the genes to retain for the concatenation; any way to subset the list in <code>x@dna</code> or <code>x@seq</code> is acceptable; by default, all genes are used.
<code>mlst</code>	if TRUE, each gene will result in a single locus in the genind object. (Default to FALSE)
<code>gapIsNA</code>	if TRUE and <code>mlst = TRUE</code> , sequences that consist entirely of gaps will be considered as NAs. (Default to FALSE)

Value

a [genind](#) object

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>, Zhian N. Kamvar, Klaus Schliep

See Also

- concatenate
- [DNABin2genind](#) to convert single DNABin objects.

Examples

```
## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
y <- multidna2multiphyDat(x)
y
z1 <- multidna2genind(x)
z1
z2 <- multiphyDat2genind(y)
all.equal(z1, z2)
```

multidna2multiphyDat *Conversions between multidna and multiphyDat*

Description

The functions `multidna2multiphyDat` and `multiphyDat2multidna` are used to convert data between [multidna](#) and [multiphyDat](#) classes.

Usage

```
multidna2multiphyDat(x)
```

```
multiphyDat2multidna(x)
```

Arguments

x a [multidna](#) or [multiphyDat](#) object.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>, Zhian N. Kamvar, Klaus Schliep

See Also

- concatenate
- [DNABin2genind](#) to convert single DNABin objects.

Examples

```
## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x

## conversion multidna -> multiplyDat
y <- multidna2multiplyDat(x)
y

## check round trip
identical(x, multiplyDat2multidna(y))
```

<code>multiplyDat-class</code>	<i>multiplyDat: class for multiple gene data</i>
--------------------------------	--

Description

This formal (S4) class is identical to [multidna](#), except that DNA sequences are stored using phyDat objects from the phangorn package. Sequences are stored as a (possibly named) list, with each element of the list being a separate DNA alignment stored as a phyDat object. The rows of the separate matrices all correspond to the same individuals, ordered identically.

Slots

`seq` a list of phyDat objects; empty slot should be NULL

`type` a character string indicating the type of the sequences stored: "DNA" for DNA sequences, "AA" for amino-acids.

`labels` a vector of labels of individuals

`n.ind` the number of individuals

`n.seq` the total number of sequences (pooling all genes), including gap sequences

`n.seq.miss` the total number of gap-only sequences

`ind.info` a data.frame containing information on the individuals, where individuals are in rows; empty slot should be NULL

`gene.info` a data.frame containing information on the genes, where genes are in rows; empty slot should be NULL

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>
 Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```

data(Laurasiatherian)

## empty object
new("multiplyDat")

## simple conversion with nicely ordered output
data(Laurasiatherian)
genes <- list(gene1=Laurasiatherian[, 1:1600],
              gene2=Laurasiatherian[, 1601:3179])
x <- new("multiplyDat", genes)
x

## trickier conversion with missing sequences / wrong order
genes <- list(gene1=Laurasiatherian[1:40,],
              gene2=Laurasiatherian[8:47,])
x <- new("multiplyDat", genes)
x

```

plot,multidna-method *Display multidna objects*

Description

Default printing for multidna objects

Usage

```

## S4 method for signature 'multidna'
plot(x, y, rows = TRUE, ask = FALSE, ...)

```

Arguments

x	a multidna object
y	an integer vector indicating the genes to plot
rows	a logical indicating if different genes should be displayed in separate rows
ask	a logical indicating if the user should be prompted between graphs
...	arguments passed to image.DNAbin

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```
## simple conversion with nicely ordered output
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
plot(x)
```

read.multidna	<i>Read multiple DNA alignments</i>
---------------	-------------------------------------

Description

These functions read multiple DNA alignments and store the output in a [multidna](#) object. They are relying on ape's original functions [read.dna](#) and [read.FASTA](#).

Usage

```
read.multidna(files, add.gaps = TRUE, ...)

read.multiFASTA(files, add.gaps = TRUE)

read.multiplyDat(files, add.gaps = TRUE, ...)
```

Arguments

files	a vector of characters indicating the paths to the files to read from.
add.gaps	a logical indicating if gap-only sequences should be added wherever sequences are missing; defaults to TRUE.
...	further arguments to be passed to the functions read.dna and read.FASTA .

Value

`read.multidna` and `read.multiFASTA` return an object of class [multidna](#), `read.multiplyDat` returns an object of class [multiplyDat](#).

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>
Klaus Schliep <klaus.schliep@gmail.com>

See Also

- [read.dna](#)
- [read.FASTA](#)
- [read.phyDat](#)

Examples

```
## get path to the files
files <- dir(system.file(package="apex"),patter="patr", full=TRUE)
files

## read files
x <- read.multiFASTA(files)
x
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(x, row=FALSE)
par(opar)

y <- read.multiplyDat(files, format="fasta")
y
```

rm.gaps

Remove gap-only sequences for missing data

Description

In [multidna](#) and [multiplyDat](#), some individuals may not be sequenced for all genes, resulting in gap-only sequences for missing data. The generic function `rm.gaps` has method for both objects; it identifies the missing sequences, and removes gap-only sequences from the alignments wherever needed.

Usage

```
rm.gaps(x, ...)

## S4 method for signature 'multidna'
rm.gaps(x, ...)

## S4 method for signature 'multiplyDat'
rm.gaps(x, ...)
```

Arguments

`x` a [multidna](#) or [multiplyDat](#) object.

`...` further arguments passed to other methods (currently not used).

show,multidna-method *Display multidna objects*

Description

Default printing for multidna objects

Usage

```
## S4 method for signature 'multidna'  
show(object)
```

Arguments

object a multidna object

Value

show returns an invisible NULL, called for side effects.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

show,multiplyDat-method *Display multiplyDat objects*

Description

Default printing for multiplyDat objects

Usage

```
## S4 method for signature 'multiplyDat'  
show(object)
```

Arguments

object a multiplyDat object

Value

show returns an invisible NULL, called for side effects.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

```
[,multidna,ANY,ANY,ANY-method
  Subset multidna objects
```

Description

Individuals in a [multidna](#) or [multiplyDat](#) object can be subsetted like the rows of a matrix, with the form `x[i,]`. Genes can be subsetted like the columns of a matrix, i.e. with the form `x[,j]`.

Usage

```
## S4 method for signature 'multidna,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'multiplyDat,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

<code>x</code>	the multidna object to subset.
<code>i</code>	a vector of logical, integers or characters to subset data by individuals; characters will be matched against individual labels.
<code>j</code>	a vector of logical, integers or characters to subset data by genes; characters will be matched against gene names labels.
<code>...</code>	further arguments to be passed to other methods; currently ignored.
<code>drop</code>	present for compatibility with the generic; currently not used.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```
data(woodmouse)
genes <- list(gene1=woodmouse[,1:500], gene2=woodmouse[,501:965])
x <- new("multidna", genes)
x
plot(x)

## keep only the first 5 individuals
x[1:5,]
plot(x[1:5,])

## keep individuals 2,4,6 and the second gene
x[c(2,4,6),2]
plot(x[c(2,4,6),2])
```

Index

[,multidna,ANY,ANY,ANY-method, [20](#)
[,multidna-method
 ([,multidna,ANY,ANY,ANY-method),
 [20](#)
[,multiplyDat,ANY,ANY,ANY-method
 ([,multidna,ANY,ANY,ANY-method),
 [20](#)
[,multiplyDat-method
 ([,multidna,ANY,ANY,ANY-method),
 [20](#)
[.multidna
 ([,multidna,ANY,ANY,ANY-method),
 [20](#)
[.multiplyDat
 ([,multidna,ANY,ANY,ANY-method),
 [20](#)

accessors, [2](#)
add.gaps, [4](#)
add.gaps,multidna-method (add.gaps), [4](#)
add.gaps,multiplyDat-method (add.gaps),
 [4](#)
add.gaps.generic (add.gaps), [4](#)
add.gaps.multidna (add.gaps), [4](#)
add.gaps.multiplyDat (add.gaps), [4](#)
as.alignment, [13](#)

concatenate, [5](#)
concatenate,multidna-method
 (concatenate), [5](#)
concatenate,multiplyDat-method
 (concatenate), [5](#)
concatenate.generic (concatenate), [5](#)
concatenate.multidna (concatenate), [5](#)
concatenate.multiplyDat (concatenate), [5](#)

data.frameOrNull (multidna-class), [11](#)
dist.dna, [6](#), [7](#)
dist.multidna, [6](#), [8](#)
DNABin2genind, [14](#)

genind, [13](#)
getLocusNames (accessors), [2](#)
getLocusNames,multidna (accessors), [2](#)
getLocusNames,multidna-method
 (accessors), [2](#)
getLocusNames,multiplyDat (accessors), [2](#)
getLocusNames,multiplyDat-method
 (accessors), [2](#)
getNumInd (accessors), [2](#)
getNumInd,multidna (accessors), [2](#)
getNumInd,multidna-method (accessors), [2](#)
getNumInd,multiplyDat (accessors), [2](#)
getNumInd,multiplyDat-method
 (accessors), [2](#)
getNumLoci (accessors), [2](#)
getNumLoci,multidna (accessors), [2](#)
getNumLoci,multidna-method (accessors),
 [2](#)
getNumLoci,multiplyDat (accessors), [2](#)
getNumLoci,multiplyDat-method
 (accessors), [2](#)
getNumSequences (accessors), [2](#)
getNumSequences,multidna (accessors), [2](#)
getNumSequences,multidna-method
 (accessors), [2](#)
getNumSequences,multiplyDat
 (accessors), [2](#)
getNumSequences,multiplyDat-method
 (accessors), [2](#)
getSequenceNames (accessors), [2](#)
getSequenceNames,multidna (accessors), [2](#)
getSequenceNames,multidna-method
 (accessors), [2](#)
getSequenceNames,multiplyDat
 (accessors), [2](#)
getSequenceNames,multiplyDat-method
 (accessors), [2](#)
getSequences (accessors), [2](#)
getSequences,multidna (accessors), [2](#)

getSequences,multidna-method
 (accessors), 2
 getSequences,multiplyDat (accessors), 2
 getSequences,multiplyDat-method
 (accessors), 2
 getTree, 7

 image.DNABin, 16
 initialize,multidna-method, 8
 initialize,multidna-methods
 (initialize,multidna-method), 8
 initialize,multiplyDat-method, 10
 initialize,multiplyDat-methods
 (initialize,multiplyDat-method),
 10

 listOrNULL (multidna-class), 11

 multidna, 2–9, 12–15, 17, 18, 20
 multidna (multidna-class), 11
 multidna-class, 11
 multidna2alignment, 12
 multidna2genind, 13
 multidna2multiplyDat, 14
 multiplyDat, 2–5, 10–14, 17, 18, 20
 multiplyDat (multiplyDat-class), 15
 multiplyDat-class, 15
 multiplyDat2alignment
 (multidna2alignment), 12
 multiplyDat2genind (multidna2genind), 13
 multiplyDat2multidna
 (multidna2multiplyDat), 14

 new.multidna
 (initialize,multidna-method), 8
 new.multiplyDat
 (initialize,multiplyDat-method),
 10

 plot,multidna-method, 16
 plot.multidna (plot,multidna-method), 16

 read.dna, 17
 read.FASTA, 17
 read.multidna, 9, 17
 read.multiFASTA (read.multidna), 17
 read.multiplyDat, 11
 read.multiplyDat (read.multidna), 17
 read.phyDat, 17
 rm.gaps, 18

 rm.gaps,multidna-method (rm.gaps), 18
 rm.gaps,multiplyDat-method (rm.gaps), 18
 rm.gaps.generic (rm.gaps), 18
 rm.gaps.multidna (rm.gaps), 18
 rm.gaps.multiplyDat (rm.gaps), 18

 setLocusNames<- (accessors), 2
 setLocusNames<-,multidna (accessors), 2
 setLocusNames<-,multidna-method
 (accessors), 2
 setLocusNames<-,multiplyDat
 (accessors), 2
 setLocusNames<-,multiplyDat-method
 (accessors), 2
 show,multidna-method, 19
 show,multiplyDat-method, 19
 show.multidna (show,multidna-method), 19
 show.multiplyDat
 (show,multiplyDat-method), 19