Package 'MultipleAlignment'

July 23, 2026

Title Representation of multiple sequence alignments in Bioconductor

Description The package implements a set of S4 classes (DNAMultipleAlignment, RNAMultipleAlignment, AAMultipleAlignment) for representing Multiple Sequence Alignments (MSA). The classes allow users to represent groups of aligned DNA, RNA or amino acid sequences as a single object. The package also provides functions to read/write such object from/to traditional MSA file formats including Stockholm and Clustal.

biocViews Alignment, MultipleSequenceAlignment, Genetics, DataImport, DataRepresentation, Infrastructure

URL https://bioconductor.org/packages/MultipleAlignment

BugReports https://github.com/Bioconductor/MultipleAlignment/issues

Version 0.99.5

License Artistic-2.0

Encoding UTF-8

Depends BiocGenerics (>= 0.59.10), S4Vectors, IRanges, Seqinfo, Biostrings

Imports methods, stats

Suggests pwalign, testthat, knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

Collate MultipleAlignment-class.R MultipleAlignment-IO.R MultipleAlignment-utils.R

git_url https://git.bioconductor.org/packages/MultipleAlignment

git_branch devel

git_last_commit fcc3e55

git_last_commit_date 2026-07-07

Repository Bioconductor 3.24

Date/Publication 2026-07-22

Author Marc Carlson [aut],
Patrick Aboyoun [aut],
Hervé Pagès [cre] (ORCID: <https://orcid.org/0009-0002-8272-4522>),
Beryl Kanali [ctb] (Converted 'MultipleAlignments' vignette from Sweave to RMarkdown),
Michael Lawrence [ctb],
Martin Morgan [ctb]

Maintainer Hervé Pagès <hpages.on.github@gmail.com>

Contents

`MultipleAlignment-class`
MultipleAlignment objects

Description

The MultipleAlignment class is a container for storing multiple sequence alignments.

Usage

```
## Constructors:
DNAMultipleAlignment(x=character(), start=NA, end=NA, width=NA,
    use.names=TRUE, rowmask=NULL, colmask=NULL)
RNAMultipleAlignment(x=character(), start=NA, end=NA, width=NA,
    use.names=TRUE, rowmask=NULL, colmask=NULL)
AAMultipleAlignment(x=character(), start=NA, end=NA, width=NA,
    use.names=TRUE, rowmask=NULL, colmask=NULL)

## ... and more (see below)
```

Arguments

`x`	Either a character vector (with no NAs), or an XString, XStringSet or XStringViews object containing strings with the same number of characters.
`start`, `end`, `width`	Either `NA`, a single integer, or an integer vector of the same length as `x` specifying how `x` should be "narrowed" (see `?narrow` in the **IRanges** package for the details).
`use.names`	`TRUE` or `FALSE`. Should names be preserved?
`rowmask`	a NormalIRanges object that will set masking for rows
`colmask`	a NormalIRanges object that will set masking for columns

Details

The MultipleAlignment class is designed to hold and represent multiple sequence alignments. The rows and columns within an alignment can be masked for ad hoc analyses.

Value

Each constructor function returns a MultipleAlignment derivative of the same class as the name of the function.

Accessor methods

In the code snippets below, x is a MultipleAlignment object.

`unmasked(x)`: The underlying XStringSet object containing the multiple sequence alignment.

`rownames(x)`: NULL or a character vector of the same length as x containing a short user-provided description or comment for each sequence in x.

`rowmask(x)`, `rowmask(x, append, invert) <- value`: Gets and sets the NormalIRanges object representing the masked rows in x. The append argument takes `union`, `replace` or `intersect` to indicate how to combine the new `value` with `rowmask(x)`. The `invert` argument takes a logical argument to indicate whether or not to invert the new mask. The `value` argument can be of any class that is coercible to a NormalIRanges via the `as` function.

`colmask(x)`, `colmask(x, append, invert) <- value`: Gets and sets the NormalIRanges object representing the masked columns in x. The append argument takes `union`, `replace` or `intersect` to indicate how to combine the new `value` with `colmask(x)`. The `invert` argument takes a logical argument to indicate whether or not to invert the new mask. The `value` argument can be of any class that is coercible to a NormalIRanges via the `as` function.

`maskMotif(x, motif, min.block.width=1, ...)`: Returns a MultipleAlignment object with a modified column mask based upon motifs found in the consensus string where the consensus string keeps all the columns but drops the masked rows.

motif The motif to mask.

min.block.width The minimum width of the blocks to mask.

... Additional arguments for `matchPattern`.

`maskGaps(x, min.fraction, min.block.width)`: Returns a MultipleAlignment object with a modified column mask based upon gaps in the columns. In particular, this mask is defined by `min.block.width` or more consecutive columns that have `min.fraction` or more of their non-masked rows containing gap codes.

min.fraction A value in `[0, 1]` that indicates the minimum fraction needed to call a gap in the consensus string (default is `0.5`).

min.block.width A positive integer that indicates the minimum number of consecutive gaps to mask, as defined by `min.fraction` (default is 4).

`nrow(x)`: Returns the number of sequences aligned in x.

`ncol(x)`: Returns the number of characters for each alignment in x.

`dim(x)`: Equivalent to `c(nrow(x), ncol(x))`.

`maskednrow(x)`: Returns the number of masked aligned sequences in x.

`maskedncol(x)`: Returns the number of masked aligned characters in x.

`maskeddim(x)`: Equivalent to `c(maskednrow(x), maskedncol(x))`.

`maskedratio(x)`: Equivalent to `maskeddim(x) / dim(x)`.

`nchar(x)`: Returns the number of unmasked aligned characters in x, i.e. `ncol(x) - maskedncol(x)`.

`alphabet(x)`: Equivalent to `alphabet(unmasked(x))`.

Coercion

In the code snippets below, x is a MultipleAlignment object.

`as(from, "DNAStringSet")`, `as(from, "RNAStringSet")`, `as(from, "AAStringSet")`, `as(from, "BStringSet")`: Creates an instance of the specified XStringSet object subtype that contains the unmasked regions of the multiple sequence alignment in x.

`as.character(x, use.names)`**:** Convert `x` to a character vector containing the unmasked regions of the multiple sequence alignment. `use.names` controls whether or not `rownames(x)` should be used to set the names of the returned vector (default is `TRUE`).

`as.matrix(x, use.names)`**:** Returns a character matrix containing the "exploded" representation of the unmasked regions of the multiple sequence alignment. `use.names` controls whether or not `rownames(x)` should be used to set the row names of the returned matrix (default is `TRUE`).

Display

The `show()` method: The letters in a MultipleAlignment object are colored when displayed by the `show()` method. Set global option `Biostrings.coloring` to FALSE to turn off this coloring.

The `detail()` method: In addition to a `show()` method, a `detail()` method is provided (`detail()` is generic function defined in the **BiocGenerics** package, see ?`detail`).

`detail(x, invertColMask, hideMaskedCols)`: Allows for a full pager driven display of the object so that masked cols and rows can be removed and the entire sequence can be visually inspected. If `hideMaskedCols` is set to its default value of `TRUE` then the output will hide all the masked colmns in the output. Otherwise, all columns will be displayed along with a row to indicate the masking status. If `invertColMask` is `TRUE` then any displayed mask will be flipped so as to represent things in a way consistent with Phylip style files instead of the mask that is actually stored in the MultipleAlignment object. Please notice that `invertColMask` will be ignored if `hideMaskedCols` is set to its default value of `TRUE` since in that case it will not make sense to show any masking information in the output. Masked rows are always hidden in the output.

Author(s)

P. Aboyoun and M. Carlson

See Also

MultipleAlignment-IO, MultipleAlignment-utils, XStringSet-class, MaskedXString-class

Examples

```
## create an object from file
origMAlign <-
  readDNAMultipleAlignment(filepath =
                           system.file("extdata",
                                       "msx2_mRNA.aln",
                                       package="MultipleAlignment"),
                           format="clustal")

## list the names of the sequences in the alignment
rownames(origMAlign)

## rename the sequences to be the underlying species for MSX2
rownames(origMAlign) <- c("Human","Chimp","Cow","Mouse","Rat",
                          "Dog","Chicken","Salmon")
origMAlign

## See a detailed pager view
if (interactive()) {
detail(origMAlign)
}
```

```
## operations to mask rows
## For columns, just use colmask() and do the same kinds of operations
rowMasked <- origMAlign
rowmask(rowMasked) <- IRanges(start=1,end=3)
rowMasked

## remove rowumn masks
rowmask(rowMasked) <- NULL
rowMasked

## "select" rows of interest
rowmask(rowMasked, invert=TRUE) <- IRanges(start=4,end=7)
rowMasked

## or mask the rows that intersect with masked rows
rowmask(rowMasked, append="intersect") <- IRanges(start=1,end=5)
rowMasked

## TATA-masked
tataMasked <- maskMotif(origMAlign, "TATA")
colmask(tataMasked)

## automatically mask rows based on consecutive gaps
autoMasked <- maskGaps(origMAlign, min.fraction=0.5, min.block.width=4)
colmask(autoMasked)
autoMasked

## cluster the masked alignments
library(pwalign)  # for stringDist()
sdist <- stringDist(as(autoMasked,"DNAStringSet"), method="hamming")
clust <- hclust(sdist, method = "single")
plot(clust)
fourgroups <- cutree(clust, 4)
fourgroups
```

`MultipleAlignment-IO` *Read/write MultipleAlignment objects*

Description

Functions to read/write MultipleAlignment objects.

Usage

```
## Read functions:
readDNAMultipleAlignment(filepath, format)
readRNAMultipleAlignment(filepath, format)
readAAMultipleAlignment(filepath, format)

## Write funtions:
write.phylip(x, filepath)
```

Arguments

x	A MultipleAlignment object
filepath	A character vector (of arbitrary length when reading, of length 1 when writing) containing the paths to the files to read or write. Note that special values like "" or "|cmd" (typically supported by other I/O functions in R) are not supported here. Also filepath cannot be a connection.
format	Either "fasta" (the default), "stockholm", "phylip", or "clustal".

Value

Each read function returns a MultipleAlignment derivative of the class that matches the name of the function.

Author(s)

P. Aboyoun and M. Carlson

See Also

MultipleAlignment

Examples

```
example(MultipleAlignment)  # make MultipleAlignment object 'autoMasked'
autoMasked

## write out the alignment object (with current masks) to Phylip format
write.phylip(autoMasked, filepath=tempfile())
```

MultipleAlignment-utils

MultipleAlignment utilities

Description

A small set of convenient utilities function to operate on a MultipleAlignment object.

Details

In the code snippets below, x is a MultipleAlignment object.

consensusMatrix(x, as.prob, baseOnly): Creates an integer matrix containing the column frequencies of the underlying alphabet with masked columns being represented with NA values. If as.prob is TRUE, then probabilities are reported, otherwise counts are reported (the default). If baseOnly is TRUE, then the non-base letters are collapsed into an "other" category.

consensusString(x, ...): Creates a consensus string for x with the symbol "#" representing a masked column. See consensusString in the **Biostrings** package for details on the arguments.

`consensusViews(x, ...)`: Similar to the `consensusString` method. It returns a XStringViews on the consensus string containing subsequence contigs of non-masked columns. Unlike the `consensusString` method, the masked columns in the underlying string contain a consensus value rather than the `"#"` symbol.

`alphabetFrequency(x, as.prob, collapse)`: Creates an integer matrix containing the row frequencies of the underlying alphabet. If `as.prob` is `TRUE`, then probabilities are reported, otherwise counts are reported (the default). If `collapse` is `TRUE`, then returns the overall frequency instead of the frequency by row.

Value

See description of each method above for what they return.

Author(s)

P. Aboyoun and M. Carlson

See Also

MultipleAlignment

Examples

```
example(MultipleAlignment)  # make MultipleAlignment object 'autoMasked'
autoMasked

## calculate frequencies
alphabetFrequency(autoMasked)
consensusMatrix(autoMasked, baseOnly=TRUE)[, 84:90]

## get consensus values
consensusString(autoMasked)
consensusViews(autoMasked)
```

Index